PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Stratégies d'accélération pour le problème de tournées de véhicules avec dépôts multiples
Auteur: Author:	Julie Mélissa Marin
Date:	2005
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Marin, J. M. (2005). Stratégies d'accélération pour le problème de tournées de véhicules avec dépôts multiples [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7643/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7643/
Directeurs de recherche: Advisors:	
Programme: Program:	Unspecified

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

STRATÉGIES D'ACCÉLÉRATION POUR LE PROBLÈME DE TOURNÉES DE VÉHICULES AVEC DÉPÔTS MULTIPLES

JULIE MÉLISSA MARIN
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
JUIN 2005

© Julie Mélissa Marin, 2005.

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-16813-4
Our file Notre référence
ISBN: 978-0-494-16813-4

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

STRATÉGIES D'ACCÉLÉRATION POUR LE PROBLÈME DE TOURNÉES DE VÉHICULES AVEC DÉPÔTS MULTIPLES

présenté par : MARIN Julie Mélissa

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. SOUMIS François, Ph.D., président

M. DESAULNIERS Guy, Ph.D., membre et directeur de recherche

M. DESROSIERS Jacques, Ph.D., membre

Le succès n'est pas au bout du chemin ; il est dans la démarche même.

REMERCIEMENTS

Je tiens d'abord à remercier mon directeur de recherche, M. Guy Desaulniers, pour son grand soutien et ses nombreux conseils prodigués tout au long de ce projet de maîtrise. Je le remercie également pour son soutien financier pendant ces deux années d'études à la maîtrise.

Je veux aussi remercier M. Ahmed Hadjar, professionnel de recherche au GERAD, pour le temps consacré à la discussion de certaines idées et pour le support informatique apporté au cours de ce projet.

Un merci spécial à M. Issmail Elhallaoui pour le soutien technique apporté pendant la partie portant sur l'agrégation dynamique de contraintes.

Sur un point de vue plus personnel, merci à mon père, Francis, pour m'avoir appris que tout est possible pour ceux qui visent haut et rêvent grand. Merci à ma mère, Yvette, pour sa compréhension, son écoute et ses nombreux encouragements toujours formulés au moment opportun. Merci à ma soeur, Marie-France, pour son incorruptible sens de l'humour qui parvient toujours à me faire oublier les petits tracas du quotidien. Merci enfin à mes amies, en particulier à Eve, pour avoir partagé avec moi les bons comme les moins bons moments de cette maîtrise, j'espère que ce n'est là que le début d'une grande amitié ; à Denise, pour croire en moi et pour m'encourager même à des milliers de kilomètres de distance ; à Sophie, pour être toujours là quand j'en ai besoin et pour notre complicité qui m'est si précieuse.

RÉSUMÉ

Ce mémoire porte sur le problème de tournées de véhicules avec dépôts multiples (PTVDM). Ce problème considère un ensemble de tâches devant être effectuées par un ensemble de véhicules répartis dans différents dépôts. Chaque tâche débute à un instant précis dans le temps. Chaque dépôt contient un nombre donné de véhicules, et tous les véhicules contenus dans un même dépôt sont du même type, c'est-à-dire qu'ils présentent tous les mêmes caractéristiques. Le PTVDM consiste donc à déterminer des tournées de véhicules à coût minimum, tout en respectant un certain nombre de contraintes. Ce problème, considéré *NP-difficile*, peut être résolu de façon exacte en utilisant une procédure de séparation et d'évaluation progressive. Le principal objectif de recherche de ce mémoire est de développer des stratégies de résolution exacte pour le PTVDM. L'impact recherché consiste, entre autres, en une diminution des temps de résolution.

Pour atteindre cet objectif, deux stratégies ont été mises de l'avant. La première reprend la méthode proposée dans l'article de HADJAR ET AL. (2003) et en modifie la partie heuristique. Cette première stratégie utilise une technique de génération de colonnes intégrée dans une procédure de séparation et d'évaluation progressive. Après avoir résolu la relaxation linéaire au noeud racine de l'arbre de branchement, une solution entière réalisable pour le PTVDM est recherchée à l'aide de la nouvelle méthode heuristique de type affectation tâche-dépôt. Une fois une solution réalisable trouvée, il faut procéder à l'élimination d'un certain nombre de variables selon un critère basé sur le coût réduit de ces variables. Le graphe sur lequel les sous-problèmes doivent être résolus devient conséquemment plus petit et cela facilite et accélère la recherche de la solution optimale du PTVDM dans la partie exacte de l'arbre de branchement. Les résultats contenus dans le chapitre 2 permettent de conclure que :

(a) La nouvelle méthode de branchement *affectation tâche-dépôt* permet d'accélérer considérablement la résolution de la partie heuristique.

(b) La qualité de la solution heuristique obtenue par la nouvelle méthode de branchement *affectation tâche-dépôt* est aussi bonne que celle obtenue, en plus de temps, par HADJAR ET AL. (2003).

(c) Au-delà d'un certain seuil, le fait d'éliminer plus de variables au noeud 0 de l'arbre de branchement n'entraîne pas nécessairement une diminution du temps total de résolution ; et inversement, c'est-à-dire le fait d'éliminer moins de variables au noeud 0 de l'arbre de branchement n'entraîne pas nécessairement une augmentation du temps total de résolution.

La seconde stratégie proposée utilise une technique d'agrégation dynamique de contraintes imbriquée dans la méthode proposée par HADJAR ET AL. (2003). La technique d'agrégation dynamique de contraintes a pour objectif de diminuer les temps de résolution des relaxations linéaires à tous les noeuds de l'arbre de branchement. L'agrégation dynamique de contraintes utilise à titre de données d'entrée une partition initiale pour le problème à résoudre. La partition initiale est obtenue à partir d'une solution réalisable générée en transformant le problème de tournées de véhicules avec dépôts multiples en un problème de tournées de véhicules à dépôt unique. Cette nouvelle variante peut être résolue en temps polynomial par des algorithmes déjà connus. Une fois la solution réalisable obtenue pour le problème de tournées de véhicules à dépôt unique, il suffit d'affecter de manière heuristique les tournées aux multiples dépôts présents dans le PTVDM. La solution heuristique ainsi obtenue permet de générer la partition initiale pour l'agrégation dynamique de contraintes. Pour cette seconde stratégie, l'effort de recherche se concentre principalement sur le

développement d'une méthode heuristique permettant d'établir une partition initiale pour l'agrégation dynamique de contraintes. Les résultats consignés dans cette partie permettent de conclure que :

(a) Le temps de résolution de la relaxation linéaire au noeud 0 de l'arbre de branchement a été réduit dans certains cas en intégrant la technique d'agrégation dynamique de contraintes à la méthode de HADJAR ET AL. (2003) pour la résolution des instances de CARPANETO ET AL. (1989).

(b) Pour des problèmes comportant le même nombre de dépôts, plus le nombre de tâches augmente, plus les améliorations en termes de temps de calcul de la relaxation linéaire sont grandes. Les améliorations les plus importantes ont été notées pour les problèmes comportant quatre dépôts, probablement à cause de la qualité de la solution heuristique permettant de générer la partition initiale.

(c) Puisque la solution de la relaxation linéaire obtenue au noeud 0 de l'arbre de branchement est presque totalement désagrégée, cela ne laisse entrevoir qu'un faible potentiel d'accélération pour la résolution complète de l'arbre de branchement. Cela justifie qu'aucun test n'a été fait pour résoudre les problèmes à l'optimalité en nombres entiers.

Une revue des écrits des méthodes utilisées, ainsi que des articles de la littérature portant sur le problème est également présentée.

ABSTRACT

This master thesis considers the multi-depot vehicle scheduling problem (MDVSP). This problem considers a set of tasks that have to be covered by a set of vehicles. All vehicles are parked in some depots located in different areas. Each task starts and ends at a given time. We make the assumption that all vehicles parked in the same depot have the same characteristics. The aim of the MDVSP is to determine schedules for the vehicles at minimum cost, while taking into account availability constraints. The MDVSP is considered as *NP-hard* but it can be solved by using a branch-and-bound procedure. The main objective of this master thesis is to develop strategies for solving exactly the MDVSP. One impact of such strategies is to reduce the computational times needed for solving this type of problem.

To reach this objective, two strategies have been developed. The first one uses the method proposed by HADJAR ET AL. (2003) and modifies its heuristic part. This method uses a column generation approach embedded into a branch-and-bound procedure. This new method works as follows. Firstly, the linear relaxation is solved at the root node of the search tree. After, the new heuristic part is applied to find a feasible solution for the MDVSP. This new heuristic is of the type "cluster-first, route-second". Hence the feasible solution is found, the "variable fixing" process is apply on the problem. This process is based on the reduced cost of the variables. The variable fixing process helps reducing the size the graph on which the sub-problems have to be solved. Consequently, it also facilitates and, sometimes, accelerates the resolution of the MDVSP to optimality. The results of the chapter 2 show that :

(a) The new heuristic method of the type "cluster-first, route-second" is faster than the one in HADJAR ET AL. (2003).

(b) The quality of the heuristic solution obtained by the new heuristic method is as good as the one obtained by HADJAR ET AL. even if the new method finds a feasible solution faster than the traditional method.

(c) Above a certain level, eliminating more variables does not necessarily yields faster solution times for the MDVSP. mean that it will At the opposite, eliminating fewer variables does not necessarily increase solution times.

The second strategy proposed consists of combining the standard method of HADJAR ET AL. (2003) with dynamic constraint aggregation . The main goal of dynamic constraint aggregation is to speed up the solution process the linear relaxations encountered throughout the search tree. An initial partition is needed as an input to dynamic aggregation method. This partition is obtained by a heuristic working as follows. First, the MDVSP is transformed into a single-depot vehicle scheduling problem (SDVSP). This new problem can be solved to optimality by using known algorithms. Once the solution is found for the SDVSP, we only need to assign heuristically the schedules to the depots of the MDVSP. The heuristic solution obtained by this procedure is used to produce the initial partition for dynamic constraint aggregation method. For this second strategy, the research effort focuses on developing the heuristic method. The results of the chapter 3 show that :

(a) The time needed to solve the linear relaxation at the root node of the search tree has been reduced for some instances of CARPANETO ET AL. (1989) by combining the dynamic constraint aggregation method with the standard method of HADJAR ET AL. (2003).

(b) For the problems containing the same number of depots, the gains in the solution time grow with the number of tasks. The instances involving four depots are the ones that present the best improvements. It is probably due to the quality of the heuristic solution.

(c) Most of the solutions obtained at the root node are completely or almost completely disaggregated. For this reason, no tests have been done for solving the instances with integrality requirements.

A literature review on the multi-depot vehicle scheduling problem and a description of the methods used in the solution process are presented.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION

Le transport urbain par autobus, subventionné en grande partie par le secteur public au Québec, subit depuis les dernières années des coupures de budget importantes imposées par les différents paliers de gouvernement. Parallèlement, l'industrie du transport de marchandises par camion évolue dans un environnement de plus en plus concurrentiel. Dans cette dynamique économique, la minimisation des coûts devient une avenue non seulement intéressante, mais vitale pour les dirigeants oeuvrant dans ces secteurs qui désirent voir leur entreprise prospérer grâce à l'accroissement de leur profit. Le transport urbain par autobus et le transport de marchandises par camion (sans contrainte de capacité) sont soumis à une réalité semblable qui peut être traduite en un problème de recherche opérationnelle bien connu, le problème de tournées de véhicules avec dépôts multiples (PTVDM).

Le problème de tournées de véhicules avec dépôts multiples (PTVDM) consiste à effectuer un ensemble de tâches en utilisant une flotte de véhicules répartis dans plusieurs dépôts, et ce en minimisant les coûts. Les domaines du transport urbain par autobus et du transport de marchandises par camion présentent différentes applications du PTVDM. Ce problème considère un ensemble de tâches $\{T_1, T_2, ..., T_n\}$. Chaque tâche T_i se définit par une heure de départ s_i, une heure d'arrivée e_i, un lieu de départ et un lieu d'arrivée. L'ensemble des tâches doit être accompli par une flotte de véhicules situés dans un des m dépôts D_k, où $k \in K = \{1, 2, ..., m\}$. Dans chaque dépôt, un nombre r_k de véhicules est disponible pour accomplir les tâches. Par hypothèse, tous les véhicules situés au même dépôt sont considérés du même type, c'est-à-dire qu'ils peuvent effectuer les mêmes tâches. En supposant qu'un dépôt physique contienne plus d'un type de véhicules, il est tout de même possible d'appliquer cette hypothèse. Dans ce cas, il suffit de séparer le dépôt physique en plusieurs dépôts distincts de manière à avoir un dépôt pour chaque type de véhicules dans le

modèle. Deux tâches peuvent être effectuées consécutivement par un même véhicule si elles sont *compatibles*. Deux tâches sont dites compatibles lorsque $s_j \geq e_i + t_{ij}$, où t_{ij} représente le temps de déplacement entre le lieu d'arrivée de la tâche T_i et le lieu de départ de la tâche T_j. Un coût c_{ij} est associé à toutes les paires de tâches T_i et T_j compatibles. Ce coût est établi en fonction de la distance à parcourir entre le lieu d'arrivée de la tâche T_i et le lieu de départ de la tâche T_j.

Les dépôts peuvent aussi être considérés comme des tâches pour former des paires compatibles. Ainsi, si un véhicule situé au dépôt D_k peut débuter (resp. terminer) sa tournée par la tâche T_j (resp. T_i), alors la paire *dépôt-tâche* (resp. *tâche-dépôt*) est dite compatible. Conséquemment, un coût c_{kj} (resp. c_{ik}), calculé de la même façon que le coût c_{ij}, est associé à chaque paire *dépôt-tâche* (resp. *tâche-dépôt*) compatible. Aussi, une pénalité établie en fonction du coût fixe d'utilisation d'un véhicule peut être ajoutée aux coûts des paires compatibles (D_k, T_j) et (T_i, D_k).

Ainsi, la fonction objectif du PTVDM consiste à minimiser :

(a) le nombre de véhicules utilisés et, par extension, les coûts fixes liés à l'utilisation de ces véhicules ;

(b) les coûts opérationnels, c'est-à-dire les coûts des déplacements à vide entre deux tâches compatibles, mais aussi entre toutes les paires *dépôt-tâche* et *tâche-dépôt* ;

(c) toutes combinaisons possibles de (a) et de (b).

Le PTVDM consiste donc à affecter un ensemble de tâches à une flotte de véhicules de manière à ce que :

(a) Chaque tâche soit effectuée par un seul véhicule ;

(b) Le nombre de véhicules quittant le dépôt k soit inférieur ou égal au nombre r_k de véhicules disponibles ;

(c) Tous les véhicules reviennent, à la fin de leur tournée, à leur dépôt initial ;

(d) La taille de la flotte de véhicules et/ou les coûts des déplacements à vide entre les tâches compatibles et les paires *dépôt-tâche* et *tâche-dépôt* soient minimisés.

Le présent travail est organisé de la façon suivante. D'abord, une revue de littérature sur le problème de tournées de véhicules avec dépôts multiples est présentée. Cette revue des écrits comprend également un bref résumé des différentes techniques de résolution utilisées dans le cadre des travaux réalisés pour ce mémoire. Le chapitre 2 montre des résultats préliminaires qui permettent de justifier l'orientation de la présente recherche. Ce même chapitre décrit la première stratégie de résolution testée ; les résultats obtenus par cette stratégie sont ensuite présentés et discutés. Le chapitre 3 expose une seconde stratégie de résolution utilisant une méthode d'agrégation dynamique de contraintes ; les résultats produits par cette méthode de résolution sont, par la suite, présentés et discutés. Pour terminer, une conclusion résume brièvement les travaux de ce mémoire tout en proposant quelques avenues de recherche potentielles pour des travaux futurs.

CHAPITRE 1 : LA REVUE DE LITTÉRATURE

1.1 Les modèles du PTVDM

Deux modèles sont couramment utilisés pour formuler le problème de tournées de véhicules à dépôts multiples (PTVDM). Il s'agit du modèle de flot multi-commodités et du modèle de partitionnement d'ensemble avec contraintes supplémentaires. Les deux sous-sections suivantes présentent la description et la formulation mathématique de ces deux modèles.

1.1.1 Le modèle de flot multi-commodités

Avant de présenter la formulation mathématique, il importe d'introduire le graphe orienté $G = (V, A)$, où V et A représentent respectivement les ensembles de noeuds et d'arcs dans le graphe G. Ce graphe est à la base du modèle de flots multi-commodités (voir, entre autres, CARPANETO ET AL. (1989)).

L'ensemble de noeuds V du graphe G se compose des tâches $\{T_1, T_2, ..., T_n\}$ et des dépôts $\{D_1, D_2, ..., D_m\}$. L'ensemble d'arcs A du graphe G se compose des arcs *intertâches*, des arcs *dépôt-tâche* et des arcs *tâche-dépôt*.

Chacune des paires de tâches T_i et T_j compatibles est reliée par m arcs *intertâches*. Les arcs *intertâches* sont notés (i, j, k) et ont une capacité égale à 1. Le coût c_{ij} d'un arc *intertâche* est relatif à la distance d_{ij} devant être parcourue entre le lieu d'arrivée de la tâche T_i et le lieu de départ de la tâche T_j.

Les arcs *dépôt-tâche* sont notés $(n + k, j, k)$ et ont une capacité égale à 1. Un arc

dépôt-tâche est placé entre un dépôt D_k et toutes les tâches T_j. Le coût $c_{n+k,j}$ de ces arcs est établi selon la fonction objectif du problème à résoudre. Ainsi, si la fonction objectif vise à minimiser les coûts opérationnels liés aux déplacements à vide, alors le coût des arcs *dépôt-tâche* se calcule selon la distance à parcourir entre le dépôt D_k et le lieu de départ de la tâche T_j. Par contre, si la fonction objectif vise à minimiser le nombre de véhicules utilisés dans la solution optimale, alors le coût des arcs $(n+k, j, k)$ pénalise l'utilisation d'un véhicule supplémentaire dans la solution optimale en supportant la totalité ou une partie du coût fixe lié à l'usage d'un véhicule. Enfin, si la fonction objectif vise à minimiser à la fois le nombre de véhicules utilisés et les coûts opérationnels liés aux déplacements à vide, alors le coût des arcs $(n+k, j, k)$ se veut une combinaison du coût de déplacement entre le dépôt D_k et le lieu de départ de la tâche T_j et du coût fixe lié à l'usage d'un véhicule.

Une des contraintes du PTVDM requiert que chaque véhicule, après avoir terminé sa tournée, retourne à son dépôt initial. Pour respecter cette contrainte, il importe donc d'ajouter au graphe G, pour chaque arc $(n+k, j, k)$, un arc de retour appelé arc *tâche-dépôt*. Ces arcs sont notés $(i, n+k, k)$ et possèdent une capacité égale à 1. Leur coût $c_{i,n+k}$ se calcule de la même façon que le coût des arcs *dépôt-tâche*.

Par contre, certains auteurs utilisent parfois une copie du dépôt pour représenter à la fois le dépôt origine et le dépôt destination dans le modèle. Dans ce cas, un premier groupe d'arcs relie le dépôt origine à toutes les tâches T_j et un second groupe d'arcs relie ces mêmes tâches au dépôt destination associé. La formulation présentée ici n'utilise pas de copie du dépôt.

Dans le modèle mathématique, une variable X_{ij}^k est associée à chaque arc $(i, j, k) \in A$. Cette variable est de type binaire et prend la valeur 1 si un véhicule provenant du dépôt D_k visite consécutivement les noeuds i et j dans une solution, où i et j $\in \{1, 2, ..., n, ..., n+k\}$. Sinon, la variable X_{ij}^k prend la valeur 0.

Aussi, l'introduction de la notation suivante est nécessaire à la compréhension du modèle mathématique

$$\delta^+(i) : \{j|(i,j,k) \in A \text{ pour au moins un } k \in K\},$$
$$\delta^-(i) : \{j|(j,i,k) \in A \text{ pour au moins un } k \in K\}.$$

La formulation mathématique du modèle de flot multi-commodités, proposée par HADJAR ET AL. (2003), s'écrit comme suit :

$$\text{Minimiser} \sum_{(i,j,k)\in A} c_{ij} X_{ij}^k \tag{1.1}$$

sujet à

$$\sum_{k\in K} \sum_{j\in\delta^+(i)} X_{ij}^k = 1 \quad \forall i = 1,2,...,n \tag{1.2}$$

$$\sum_{j=1}^{n} X_{n+k,j}^k \leq r_k \quad \forall k \in K \tag{1.3}$$

$$\sum_{j\in\delta^-(i)} X_{ji}^k - \sum_{j\in\delta^+(i)} X_{ij}^k = 0 \quad \forall k \in K,\ i = 1,2,...,n,n+k \tag{1.4}$$

$$X_{ij}^k \in \{0,1\} \quad \forall (i,j,k) \in A. \tag{1.5}$$

Les contraintes (1.2) veillent à ce que chaque tâche soit effectuée par un et un seul véhicule. Les contraintes (1.3) s'assurent que le nombre de véhicules partant du dépôt D_k soit inférieur ou égal au nombre de véhicules disponibles dans ce même dépôt. Enfin, les contraintes (1.4) vérifient la conservation de flot pour tous les noeuds du graphe, tâches et dépôts confondus. La fonction objectif (1.1) a pour but de minimiser à la fois le nombre de véhicules utilisés dans la solution optimale et les coûts opérationnels liés aux déplacements à vide. La solution optimale de ce modèle fournit le nombre de véhicules utilisés, les tâches effectuées par chacun de ces véhicules et le coût total relatif à cette solution.

1.1.2 Le modèle de partitionnement d'ensemble

La deuxième formulation possible est un modèle de partitionnement d'ensemble avec contraintes supplémentaires. Cette formulation diminue le nombre de contraintes à traiter dans le problème à résoudre. Par contre, le nombre de variables peut exploser de façon exponentielle étant donné que ces dernières ne sont plus associées aux arcs, mais plutôt aux *tournées* possibles. Dans ce cas-ci, une tournée p se définit comme un ensemble d'arcs. En d'autres termes, une tournée se veut une *charge de travail*, c'est-à-dire un regroupement de tâches pouvant être accompli par un même véhicule. Le coût de cette tournée, noté c_p, est égal à la somme des coûts associés aux arcs formant cette tournée. La formulation par partitionnement d'ensemble requiert donc que toutes les tournées possibles, c'est-à-dire l'ensemble Ω, soient connues à l'avance. Aussi, des constantes binaires, a_{ip} et b_p^k, sont nécessaires à la formulation du modèle de partitionnement d'ensemble. La constante a_{ip} prend la valeur 1 si et seulement si la tournée p inclut la tâche T_i ; sinon, elle prend la valeur 0. La constante b_p^k prend la valeur 1 si et seulement si la tournée p débute au dépôt k ; sinon, elle prend la valeur 0.

Le modèle de RIBEIRO ET SOUMIS (1994) s'écrit comme suit :

$$\text{Minimiser} \sum_{p \in \Omega} c_p \theta_p \tag{1.6}$$

sujet à

$$\sum_{p \in \Omega} a_{ip} \theta_p = 1 \qquad \forall i = 1, 2, ..., n \tag{1.7}$$

$$\sum_{p \in \Omega} b_p^k \theta_p \leq r_k \qquad \forall k = 1, 2, ..., m \tag{1.8}$$

$$\theta_p \in \{0, 1\} \qquad \forall p \in \Omega. \tag{1.9}$$

Les contraintes (1.7) et (1.8) ont les mêmes fonctions que les contraintes (1.2) et (1.3) du modèle de flot multi-commodités. Les contraintes de conservation de flot sont véri-

fiées implicitement puisqu'elles ont été considérées lors de la formation de l'ensemble Ω. Chaque variable binaire θ_p est associée à une tournée réalisable. La variable prend la valeur 1 si la tournée p fait partie de la solution. Sinon, elle prend la valeur 0. La fonction objectif (1.6) vise toujours à minimiser le nombre de véhicules utilisés dans la solution optimale ainsi que les coûts opérationnels liés aux déplacements à vide. Bref, le modèle de partitionnement d'ensemble (1.6)-(1.9) est équivalent au modèle de flot multi-commodités (1.1)-(1.5) puisqu'il peut être obtenu de ce dernier en appliquant le principe de décomposition de Dantzig-Wolfe (voir DANTZIG ET WOLFE (1960)).

1.2 La revue des écrits

Le problème de tournées de véhicules à dépôts multiples (PTVDM) est un problème classique en recherche opérationnelle. En fait, plusieurs chercheurs ont travaillé à élaborer des méthodes exactes ou heuristiques pour obtenir une solution à ce problème. Cette revue des articles n'a pas la prétention de recenser la totalité des écrits sur ce problème. Elle tente plutôt d'exposer une vue d'ensemble des travaux les plus pertinents dans le cadre de cette recherche.

Étant donné que ce mémoire s'intéresse davantage à la méthode de résolution qu'à la modélisation du PTVDM, la revue de littérature est présentée selon les deux grands types d'approches utilisées pour la résolution : les approches heuristiques et les approches exactes.

1.2.1 Les approches heuristiques

BERTOSSI ET AL. (1987) sont les premiers à montrer que le PTVDM est NP-complet pour $|K| \geq 2$. Les auteurs formulent le PTVDM comme un problème de couplage

maximal de flot multi-commodités avec contraintes de capacité. L'approche heuristique qu'ils proposent repose sur l'utilisation d'une relaxation lagrangienne du problème original. Cette approche consiste à résoudre un problème d'affectation des véhicules aux tâches pour chaque dépôt. La relaxation lagrangienne permet donc d'obtenir une borne inférieure pour le problème original. Cependant, comme le PTVDM est résolu une fois pour chaque dépôt, cela implique qu'une tâche peut être affectée à plus d'un dépôt, ce qui entraîne la violation des contraintes (1.2) du modèle de flot multi-commodités. Conséquemment, la solution de la relaxation lagrangienne n'est pas nécessairement réalisable. C'est pourquoi un algorithme glouton génère, par la suite, une solution réalisable pour le problème original en éliminant de la solution de la relaxation lagrangienne toutes les violations de contraintes. Les auteurs présentent des résultats pour des problèmes comportant jusqu'à 50 tâches et trois dépôts.

MESQUITA ET PAIXÃO (1992) et (1994) ont également étudié le PTVDM en utilisant un modèle de flot multi-commodités et une approche heuristique basée sur une relaxation lagrangienne. Les instances résolues par leur approche comptent jusqu'à 400 tâches et six dépôts.

En 1993, DELL'AMICO ET AL. utilisent la théorie des graphes pour modéliser le PTVDM. La méthode heuristique proposée par ces auteurs consiste à résoudre un problème de plus court chemin dans un graphe. À chaque itération, un algorithme détermine une charge de travail pour un nouveau véhicule et un ensemble d'arcs interdits. La détermination d'un ensemble d'arcs interdits, à chaque itération, permet d'obtenir une solution réalisable minimisant le nombre de véhicules utilisés. La méthode heuristique est suivie d'une procédure de raffinement composée de quatre étapes. D'abord, une réaffectation des tâches couvertes par les véhicules d'un même dépôt est effectuée. Par la suite, les tâches comprises à l'intérieur d'une même charge de travail sont échangées. La troisième étape consiste à recenser toutes les paires de charges de travail constituées de tâches couvertes par plus d'un dépôt. Ces tâches

sont ensuite réaffectées aux deux véhicules de la paire de manière à respecter la contrainte voulant que chaque tâche soit couverte par un seul véhicule. La dernière étape a pour but de réaffecter optimalement les charges de travail aux véhicules. La méthode heuristique proposée par ces auteurs combinée aux procédures de raffinement tout juste énoncées a permis de résoudre des problèmes comptant jusqu'à 1000 tâches et dix dépôts. Cependant, la qualité de la solution produite par la méthode heuristique diminue au fur et à mesure que le nombre de dépôts augmente. La procédure de raffinement semble être bénéfique à appliquer puisque son temps de calcul est négligeable par rapport à l'amélioration qu'elle produit sur la solution heuristique initialement trouvée. L'approche proposée par DELL'AMICO ET AL. garantit l'utilisation du nombre minimal de véhicules.

1.2.2 Les approches exactes

CARPANETO ET AL. (1989) présentent la première méthode exacte pour résoudre le PTVDM. Ces auteurs utilisent un modèle de flot multi-commodités pour formuler le PTVDM. Leur approche repose sur l'utilisation d'une procédure de séparation et d'évaluation progressive. À chaque noeud de l'arbre, la borne inférieure est obtenue en résolvant, entre autres, un problème d'affectation compris à l'intérieur d'une procédure additive. Les auteurs révèlent avoir résolu à l'optimalité des problèmes ayant jusqu'à 60 tâches et trois dépôts.

RIBEIRO ET SOUMIS (1994) modélisent quant à eux le PTVDM en utilisant un modèle de partitionnement d'ensemble avec contraintes supplémentaires. Ils proposent une approche de génération de colonnes pour résoudre la relaxation linéaire du PTVDM. La résolution de la relaxation linéaire permet d'obtenir une borne inférieure pour le problème original. Le sous-problème consiste à résoudre un problème de plus court chemin, c'est-à-dire partir du dépôt D_k, visiter un ou plusieurs noeuds

de type *tâche* et revenir au dépôt D_k initial. Les auteurs montrent que leur borne inférieure est aussi bonne que celle obtenue par la méthode additive proposée par CARPANETO ET AL. (1989). Dans le cas où la solution obtenue par la relaxation linéaire est non-entière, alors l'utilisation d'une procédure de séparation et d'évaluation progressive permet de générer une solution entière. La stratégie de branchement utilisée est de type *profondeur d'abord*. À la lumière des résultats présentés dans RIBEIRO ET SOUMIS (1994), il est possible de souligner que la méthode de génération de colonnes est robuste. En fait, les temps de calcul demeurent raisonnables même lorsque le nombre de dépôts augmente. De plus, la qualité de la borne inférieure ne se détériore pas avec l'augmentation de la taille des problèmes. Enfin, la qualité de la borne inférieure permet de résoudre à l'optimalité des problèmes comportant jusqu'à 300 tâches et six dépôts.

Une autre méthode exacte est présentée par FORBES ET AL. (1994). Celle-ci se divise en trois étapes. La première étape consiste à résoudre une relaxation du PTVDM, c'est-à-dire un problème de quasi-affectation. Deuxièmement, le modèle utilisé à la première étape est modifié pour tenir compte des contraintes de conservation de flot pour tous les noeuds du graphe et de l'ajustement de la fonction objectif pour que celle-ci minimise à la fois le nombre de véhicules utilisés et les coûts opérationnels liés aux déplacements à vide. La relaxation linéaire du problème est résolue par la méthode du simplexe dual en utilisant comme solution initiale la solution trouvée à la première étape. La troisième et dernière étape est mise en oeuvre seulement lorsque la solution obtenue à la deuxième étape n'est pas entière. Dans ce cas, il faut appliquer une procédure de séparation et d'évaluation progressive pour obtenir une solution entière. Les résultats présentés dans cet article montrent que la relaxation linéaire génère une excellente borne inférieure. La relaxation linéaire constitue donc un bon point de départ pour l'application d'une procédure de séparation et d'évaluation progressive. Les auteurs rapportent des résultats pour des problèmes présentant jusqu'à 600 tâches et trois dépôts.

BIANCO ET AL. (1994) utilisent une formulation semblable à celle de RIBEIRO ET SOUMIS (1994) puisqu'ils emploient un modèle de partitionnement d'ensemble avec contraintes supplémentaires. BIANCO ET AL. présentent une méthode heuristique pour résoudre le problème dual de la relaxation linéaire du modèle de partitionnement d'ensemble. Cette méthode consiste en une série d'approximations successives. Ces auteurs innovent en introduisant un principe qui permet de fixer des variables. Cela consiste à attribuer une valeur nulle à toutes les variables identifiées comme ne pouvant appartenir à la solution optimale entière. Ces variables sont donc dites *fixées* ou *éliminées.* Le principe permettant de fixer des variables a pour effet de diminuer la taille du problème à résoudre en en réduisant le nombre de variables. Lorsqu'une solution réalisable, mais non nécessairement optimale est trouvée, les auteurs proposent de résoudre le problème à l'optimalité en utilisant une procédure de séparation et d'évaluation progressive. La méthode proposée par BIANCO ET AL. (1994) est plus rapide que celle de RIBEIRO ET SOUMIS (1994) et peut résoudre des problèmes comportant jusqu'à 300 tâches et six dépôts.

Dans un article publié en 1995, MINGOZZI ET AL. reprennent la formulation et l'approche proposées dans BIANCO ET AL. (1994) pour optimiser les horaires d'une flotte de véhicules dans le but de satisfaire les commandes d'un groupe de clients pendant une période de temps donnée. Dans cet article, les auteurs soulèvent un cas plus général du PTVDM, le problème de tournées de véhicules avec *fenêtres de temps* et dépôts multiples (PTVFTDM). Une fenêtre de temps se définit comme l'intervalle de temps pendant lequel une tâche doit débuter. Les auteurs cités jusqu'à présent ont étudié le cas particulier de la fenêtre de temps de largeur nulle. Lorsque la fenêtre de temps est de largeur nulle, cela signifie que la tâche doit débuter à un instant précis dans le temps et non à l'intérieur d'une période donnée. MINGOZZI ET AL. tentent de résoudre ce problème plus général en utilisant une adaptation de l'approche proposée dans l'article de BIANCO ET AL. (1994). Ils rapportent des résultats pour des problèmes comptant jusqu'à 120 tâches et cinq dépôts.

DESAULNIERS ET AL. (1998) continuent d'explorer le problème de tournées de véhicules avec fenêtres de temps et dépôts multiples (PTVFTDM). Ils formulent le problème en utilisant un modèle de flot multi-commodités en nombres entiers. Ces auteurs étudient un nouvel aspect en utilisant les coûts exacts pour modéliser l'attente au lieu d'utiliser les coûts minimaux, comme les autres auteurs le faisaient auparavant. La méthode de résolution proposée consiste en la combinaison d'une méthode de génération de colonnes et d'une procédure de séparation et d'évaluation progressive. En d'autres termes, DESAULNIERS ET AL. (1998) généralisent l'approche proposée dans RIBEIRO ET SOUMIS (1994) pour l'adapter à un cas plus général du PTVDM, c'est-à-dire le cas avec fenêtres de temps de largeur non-nulle (PTVFTDM). Les auteurs ont testé leur approche sur deux types de problèmes : le problème de planification d'horaires d'autobus urbains et le problème de planification d'horaires de camions destinés au transport des marchandises. Ces deux types de problèmes ont été générés selon la méthode présentée dans CARPANETO ET AL. (1989). Comme ce mémoire s'intéresse plus particulièrement au transport urbain, les résultats rapportés ici sont ceux concernant ce type de problème. D'abord, les problèmes qui tiennent compte du coût exact d'attente entre deux tâches sont plus difficiles à résoudre que ceux utilisant le coût minimal d'attente puisque le premier type de problème nécessite une ressource supplémentaire pour la résolution des sous-problèmes. Aussi, l'écart d'intégrité est relativement petit pour la plupart des problèmes étudiés. Cela signifie que le nombre de véhicules utilisés est relativement le même dans la solution de la relaxation linéaire et dans la solution en nombres entiers. Un pourcentage d'erreur moyen de seulement 3% a été relevé pour le coût total d'opération lorsque le coût exact d'attente est utilisé au lieu du coût minimal d'attente. Cela amène donc les auteurs à conclure que l'utilisation du coût exact d'attente n'est pas toujours justifiable étant donné que son approximation par le coût minimal d'attente est assez juste. Par contre, cette dernière conclusion aurait pu être différente si les fenêtres de temps utilisées avaient été plus étendues étant donné que le pourcentage moyen

d'erreur tend à augmenter quand la largeur de la fenêtre de temps augmente. Aussi, les résultats présentés dans DESAULNIERS ET AL. (1998) montrent que les temps de calcul n'augmentent pas nécessairement avec le nombre de dépôts. Les auteurs présentent des résultats pour des instances ayant jusqu'à 600 tâches et cinq dépôts.

Plus récemment, des travaux effectués par LÖBEL (1998) ont permis de résoudre des problèmes d'une taille beaucoup plus imposante. Ces problèmes comptent jusqu'à 25000 tâches et 49 dépôts. Une des raisons qui explique que LÖBEL puisse résoudre des problèmes de cette taille est qu'il utilise une version du PTVDM comprenant des contraintes de restriction tâche-dépôt. Cela signifie que chaque tâche ne peut être accomplie que par un groupe restreint de dépôts. Il est donc possible de fixer plusieurs variables dès le début et la taille du problème se trouve ainsi considérablement réduite. La structure particulière des problèmes de LÖBEL est une autre raison qui justifie la taille impressionnante des instances que ce chercheur résout. Dans son modèle, LÖBEL définit une tâche comme étant une partie de tournée de véhicules, contrairement au modèle standard qui définit une tâche comme étant une tournée de véhicules. Entre chaque partie d'une même tournée, les distances sont nulles tandis qu'elles sont positives vers les autres tâches. Dans ce cas, il est donc fort probable que les tâches provenant d'une même tournée demeurent ensemble dans la solution optimale. LÖBEL propose un modèle de flot multi-commodités en nombres entiers pour formuler le PTVDM. L'approche proposée par cet auteur consiste à utiliser deux types de relaxation lagrangienne et de résoudre à l'aide d'une technique de génération de colonnes. D'abord, l'auteur propose de résoudre un programme linéaire restreint du programme original en nombres entiers. Pour résoudre cette relaxation, deux méthodes heuristiques sont proposées. Il s'agit de *schedule-cluster-reschedule* (*SCR*) et de *nearest depot* (*ND*). Les méthodes heuristiques utilisées montrent que la méthode *SCR* produit de meilleures solutions que la méthode *ND*, mais cette dernière est par contre plus rapide. Pour chaque dépôt, une deuxième résolution est effectuée pour

réaffecter toutes les tâches qui ont été attribuées de manière heuristique à ce dépôt. Troisièmement, il s'agit de résoudre la relaxation linéaire en utilisant une technique de génération de colonnes. Les sous-problèmes consistent à résoudre un problème de flot à coût minimum. La génération de colonnes s'effectue en deux phases. Premièrement, la phase dite lagrangienne est appliquée tant et aussi longtemps que la valeur de la fonction objectif diminue entre deux programmes linéaires restreints. La diminution est gérée par un seuil de tolérance préalablement défini. La deuxième phase est en fait la phase standard de génération de colonnes. Dans cette phase, chaque programme linéaire restreint utilise comme base initiale la dernière base du programme linéaire restreint précédent. Cette phase s'arrête seulement lorsque l'optimalité du programme linéaire restreint peut être prouvée optimale pour le programme complet. Une des particularités de la méthode proposée par LÖBEL réside dans le fait que le processus de génération de colonnes engendre des arcs intertâches, c'est-à-dire des variables X_{ij}^k, plutôt que des tournées, c'est-à-dire des colonnes θ_p, comme c'était le cas jusqu'à présent pour les autres auteurs utilisant la technique de génération de colonnes pour solutionner le PTVDM. La méthode de LÖBEL a fait ses preuves quant à la taille des instances qu'elle peut résoudre. Aussi, la qualité de la borne trouvée grâce à la relaxation linéaire est bonne. En fait, de manière générale, la relaxation linéaire engendre la solution optimale entière du problème à résoudre. Si toutefois ce n'est pas le cas, les résultats de LÖBEL montrent que la solution optimale entière du problème à résoudre peut être obtenue en arrondissant à l'entier supérieur la solution optimale de la relaxation linéaire. Enfin, la méthode de LÖBEL permet à la partie ayant le plus grand poids dans la fonction objectif, c'est-à-dire la grosseur de la flotte de véhicules, de converger très rapidement à sa valeur minimale.

Une nouvelle approche dite polyhédrale est proposée par FISCHETTI ET AL. (2001). Ces auteurs formulent le PTVDM en un programme linéaire en nombres binaires. Dans cette formulation, une variable est associée à chaque *intertâche* possible (deux

tâches compatibles résultant en une intertâche possible). Le modèle est ensuite résolu en utilisant des familles d'inégalités valides pour lesquelles des méthodes de séparation exactes et heuristiques sont proposées. Ces méthodes sont combinées à une procédure de séparation et d'évaluation progressive s'appuyant sur une stratégie de type *meilleur d'abord.* Les familles d'inégalités proposées permettent notamment d'éliminer toutes les tournées ne contenant aucun dépôt, ou toutes celles contenant deux dépôts différents, c'est-à-dire que le dépôt initial et le dépôt final ne sont pas les mêmes. Grâce à leur approche, FISCHETTI ET AL. (2001) ont pu résoudre des instances comptant jusqu'à 300 tâches et cinq dépôts.

1.2.3 La méthode de base pour ce mémoire

Le dernier article cité est aussi celui qui constitue la base des stratégies de résolution développées au chapitre 2 et 3 de ce mémoire. C'est pourquoi il est présenté plus en détails dans les lignes qui suivent. HADJAR ET AL. (2003) ont récemment exposé une nouvelle méthode exacte pour résoudre des instances de grande taille du PTVDM. Ces auteurs utilisent un modèle de flot multi-commodités pour formuler le PTVDM. Dans ce modèle, chaque variable binaire est associée à un arc (i, j, k). Pour résoudre ce modèle, les auteurs proposent de combiner quatre méthodes : une procédure de séparation et d'évaluation progressive, une procédure permettant de fixer les variables, une technique de génération de colonnes et une technique de génération de contraintes. L'approche utilisée par ces auteurs est de type « branch-and-cut » et se divise en deux parties : *la partie heuristique* et *la partie exacte.*

La partie heuristique a pour but de trouver une solution entière réalisable, mais non nécessairement optimale, pour le problème à résoudre. Dans un premier temps, la relaxation linéaire du programme original est résolue par génération de colonnes au noeud racine de l'arbre de branchement. La solution duale de la relaxation linéaire

et la valeur associée à cette solution sont notées respectivement $\bar{\pi}$ et Z_{LP}. Par la suite, la recherche d'une solution entière passe par l'application d'une procédure heuristique de séparation et d'évaluation progressive privilégiant une stratégie de type *profondeur d'abord*, qui ne permet pas de retour en arrière. À chaque noeud de l'arbre de branchement, des plans coupants sont recherchés (voir la partie 1.2.3.1 pour la description des plans coupants). Lorsque des plans coupants sont trouvés, alors un noeud-fils est créé en ajoutant ces plans coupants aux contraintes déjà actives au noeud-père. Si aucun plan coupant n'est trouvé, alors des arcs intertâches sont fixés pour créer le noeud-fils. Les arcs intertâches sont placés en ordre décroissant selon la valeur du flot passant sur chaque arc, c'est-à-dire que l'arc ayant le flot le plus élevé est placé en tête de liste, et ainsi de suite jusqu'à celui supportant le plus petit flot. Par la suite, un groupe de paires de tâches (i, j), noté A_P, est choisi tel que $|A_P| \geq 1$ et

$$\sum_{(i,j)\in A_P} \left(1 - \sum_{k=1}^{|K|} X_{ij}^k\right) \leq ComplementSumMax$$

où *ComplementSumMax* est un paramètre prenant la valeur 0,75. Les arcs sont sélectionnés en respectant l'ordre de la liste établie préalablement. La création du noeud-fils consiste en l'ajout de la contrainte $\sum_{k=1}^{|K|} X_{ij}^k = 1 \ \forall (i,j) \in A_P$ aux contraintes déjà actives au noeud-père. La procédure de séparation et d'évaluation progressive se poursuit ainsi jusqu'à ce qu'une solution entière soit trouvée. La valeur de la solution entière ainsi obtenue est notée Z_{IP}.

Une fois la solution entière obtenue par la résolution de la partie heuristique, l'exploration de la partie exacte peut débuter. Pour ce faire, il faut retourner au noeud 0 et attribuer une valeur nulle à toutes les variables qui ont un coût réduit tel que $\bar{c}_{ij}^k \geq Z_{IP} - Z_{LP}$, où $\bar{c}_{ij}^k$ désigne le coût réduit de la variable X_{ij}^k en fonction de $\bar{\pi}$. En d'autres termes, toutes les variables ayant un coût réduit supérieur ou égal à la différence entre la valeur de la solution entière trouvée dans la partie heuristique et la

valeur de la solution duale de la relaxation linéaire obtenue au noeud 0 doivent être éliminées puisqu'elles ne peuvent faire partie de la solution optimale entière. Cela permet de réduire la taille du graphe sur lequel il faut résoudre les sous-problèmes, mais aussi de diminuer le nombre de colonnes contenues dans le problème maître puisque toutes les colonnes qui contiennent un arc ayant un flot égal à 0 sont exclues du problème maître. Conséquemment, la procédure de séparation et d'évaluation progressive est appliquée sur un problème de taille plus raisonnable.

Dans la partie exacte, deux sortes de branchement sont possibles à chaque noeud : (i) le branchement par partitionnement de couleurs ; (ii) le branchement par intertâches fixées.

(i) Le branchement par partitionnement de couleurs consiste à séparer les dépôts en deux groupes K_1^i et K_2^i tels que $|K_1^i|$ et $|K_2^i|$, dans un premier temps, et $\sum_{k \in K_1^i} \sum_j X_{ij}^k$ et $\sum_{k \in K_2^i} \sum_j X_{ij}^k$ pour une tâche i donnée, dans un deuxième temps, soient les plus proches possibles. Il faut ensuite chercher un i tel que $\sum_{k \in K_1^i} \sum_j X_{ij}^k$ et $\sum_{k \in K_2^i} \sum_j X_{ij}^k$ soient compris entre 0,4 et 0,6 tout en s'assurant que le nombre de dépôts k tel que $X_{ij}^k > 0$ (pour tout j) soit maximisé. Si un tel i n'existe pas, il faut passer au type de branchement suivant. Par contre, si un tel i existe, alors le branchement s'effectue comme suit. Le noeud-fils de droite (resp. de gauche) supporte l'hypothèse voulant que $\sum_{k \in K_1^i} \sum_j X_{ij}^k = 0$ (resp. $\sum_{k \in K_2^i} \sum_j X_{ij}^k = 0$). Cela a pour effet de réduire la taille du graphe en éliminant tous les arcs (i, j, k) pour $k \in K_1^i$ (resp. $k \in K_2^i$) si la branche de droite (resp. de gauche) est choisie.

(ii) Le branchement par intertâches fixées consiste à choisir une paire (i, j) telle que $\sum_{k=1}^{|K|} X_{ij}^k$ est la plus proche possible de 0,5. Par la suite, deux noeuds-fils sont créés en ajoutant aux contraintes du noeud-père la contrainte suivante pour le noeud-fils de droite (resp. de gauche) : $\sum_{k=1}^{|K|} X_{ij}^k = 1$ (resp. $\sum_{k=1}^{|K|} X_{ij}^k = 0$).

Chaque noeud de la partie exacte est donc créé par une de ces deux méthodes de

branchement à laquelle il est possible d'ajouter des plans coupants du même type que ceux décrits dans la partie 1.2.3.1.

En résumé, la partie exacte de l'arbre de branchement met en oeuvre une stratégie de type *meilleur en profondeur d'abord* (best-then-depth search). Selon cette stratégie, le noeud sur lequel il faut effectuer le branchement est celui qui présente la plus petite borne inférieure (critère de meilleur d'abord). Dans le cas où plusieurs noeuds présentent la même borne inférieure, il faut brancher sur le noeud créé par l'ajout d'une contrainte du type $\sum_{k=1}^{|K|} X_{ij}^k = 1$. Une fois le noeud choisi selon le critère de la meilleure borne inférieure, l'arbre de branchement est exploré selon une stratégie de type *profondeur d'abord* en choisissant les contraintes de la forme $\sum_{k=1}^{|K|} X_{ij}^k = 1$ (si le noeud actuel a été créé par le branchement par intertâches fixées) ou $\sum_{k \in K_1^i} \sum_j X_{ij}^k = 0$ (si le noeud actuel a été créé par le branchement par partitionnement de couleurs). Lorsqu'un noeud est élagué pendant l'exploration de type profondeur d'abord, l'algorithme retourne à l'exploration de type meilleur d'abord. Les avantages relevés par HADJAR ET AL. (2003) pour justifier l'utilisation de cette stratégie est que l'algorithme converge rapidement vers une solution entière et que la solution du problème maître, à un noeud-fils donné, peut être trouvée plus vite en utilisant comme solution initiale la solution optimale trouvée au noeud-père.

De manière générale, chaque noeud compris dans la partie heuristique (resp. la partie exacte), à l'exception du noeud 0, possède un (resp. deux) noeud-fils. Le noeud 0 possède, quant à lui, trois noeuds-fils, c'est-à-dire un noeud-fils appartenant à la partie heuristique et deux noeuds-fils appartenant à la partie exacte.

1.2.3.1 Les plans coupants

HADJAR ET AL. (2003) présentent dans leur article plusieurs inégalités valides qu'il est possible d'inclure comme plans coupants dans la procédure de séparation et d'évaluation progressive. La présente section donne un exemple d'un de ces plans coupants.

Il faut, dans un premier temps, introduire le concept de *sous-multigraphe avec épine*, duquel il est possible de générer un plan coupant. HADJAR ET AL. définissent le sous-multigraphe avec épine comme suit :

> Soit $H = (S, F)$ un sous-multigraphe de G. Il est possible de dire que H est un sous-multigraphe avec épine si S peut être divisé en trois groupes, S_1, S_2 et S_3, et F peut être divisé en deux groupes, F_1 et F_2, tel que :
>
> (a) $S_1 \cup S_2$ ne contient aucun dépôt ;
>
> (b) $|S_1|$ est impair et d'au moins trois ;
>
> (c) $\delta^-_{F_2}(i) = \emptyset$ et $coul$[1] $(\delta^-_{F_1}(i)) \cap coul(\delta^+_{F_1}(i)) = \emptyset$, pour tout $i \in S_1$;
>
> (d) $coul(\delta^-_{F_2}(i)) \cap coul(\delta^-_{F_1}(i)) = \emptyset$ et $coul(\delta^-_{F_2}(i)) \cap coul(\delta^+_{F_1}(i)) = \emptyset$, pour tout $i \in S_2$;
>
> (e) $\delta^-_F(i) = \emptyset$, $\delta^+_F(i) \subseteq F_2$ et $\delta^+_F(i) \neq \emptyset$, pour tout $i \in S_3$.

À partir de cette définition, les auteurs dégagent une inégalité valide pouvant être utilisée à titre de plan coupant.

> Soit $H = (S, F)$ un sous-multigraphe avec épine de G, où $S = S_1 \cup S_2 \cup S_3$ et $F = F_1 \cup F_2$. L'inégalité $\sum_{(i,j,k)\in F} X^k_{ij} \leq \lfloor |S_1|/2 \rfloor + |S_2|$ est donc valide pour le programme linéaire en nombres entiers (1.1)-(1.5).

Afin d'illustrer le plan coupant introduit plus haut, il importe de se référer à la figure 1.1.

[1]Le terme *coul* fait référence à la couleur de l'arc ; celle-ci est déterminée par la couleur du dépôt couvrant l'arc en question. Dans la figure 1.1, les couleurs sont représentées par les différentes textures (arc plein (D_2), arc pointillé pâle (D_3), arc pointillé foncé (D_1)).

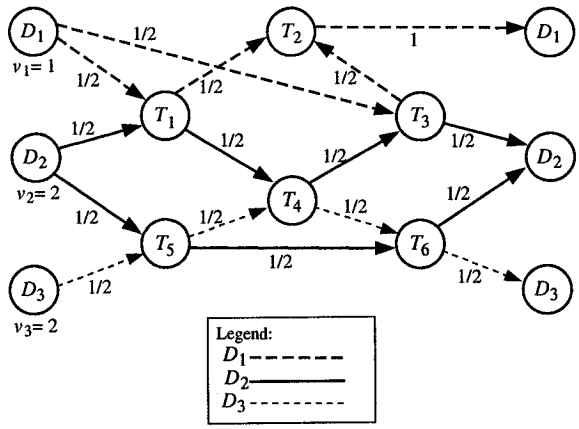

Figure 1.1 – Sous-multigraphe avec épine - Tiré de HADJAR ET AL. (2003)

Dans cette figure, le sous-multigraphe avec épine $H = (S, F)$, où $S = S_1 \cup S_2 \cup S_3$ et $F = F_1 \cup F_2$ se compose comme suit :

$$S_1 = T_1, T_2, T_3;$$

$$S_2 = T_4;$$

$$S_3 = T_5;$$

$$F_1 = (T_1, T_2), (T_1, T_4), (T_3, T_2), (T_4, T_3);$$

$$F_2 = (T_5, T_4).$$

En revoyant chaque condition qui définisse un sous-multigraphe avec épine, il est possible de conclure que :

(a) $S_1 \cup S_2$ ne contient aucun dépôt ;

(b) $|S_1|$ est impair et d'au moins trois ;

(c) Pour tous les noeuds i appartenant à l'ensemble S_1, (i) aucun des arcs prédécesseurs de i n'est compris dans F_2, et (ii) les couleurs des arcs prédécesseurs et des arcs successeurs de i compris dans F_1 sont différentes ;

(d) Pour tous les noeuds i appartenant à l'ensemble S_2, (i) les couleurs des arcs prédécesseurs compris dans F_1 et dans F_2 sont différentes, et (ii) les couleurs de l'arc prédécesseur compris dans F_2 et de l'arc successeur compris dans F_1 sont différentes ;

(e) Pour tous les noeuds i appartenant à l'ensemble S_3, (i) aucun arc prédécesseur de i n'est compris dans le sous-multigraphe avec épine H, et (ii) l'arc successeur de i est compris dans F_2, et (iii) i possède un arc successeur.

De cet exemple, il est possible de déduire l'inégalité valide ou le plan coupant suivant :

$$X_{T_1,T_2} + X_{T_1,T_4} + X_{T_3,T_2} + X_{T_4,T_3} + X_{T_5,T_4} \leq \lfloor |3|/2 \rfloor + |1| = 2$$

Le plan coupant décrit dans cette partie n'est qu'un exemple parmi tous ceux énoncés dans l'article de HADJAR ET AL. (2003). Le lecteur désirant en savoir davantage à ce propos peut se référer à l'article en question.

1.3 Les techniques utilisées

1.3.1 La méthode de génération de colonnes

La génération de colonnes est une méthode de décomposition utilisée pour résoudre des problèmes de grande taille comptant un très grand nombre de variables. À chaque itération, la méthode du simplexe résout un programme linéaire restreint (appelé problème maître restreint) composé d'un sous-ensemble de variables et un algorithme, généralement de complexité polynomiale ou pseudo-polynomiale, solutionne un sous-problème qui permet de générer implicitement toutes les variables non-considérées dans le problème maître restreint. Ces variables sont aussi appellées colonnes. La résolution du problème maître restreint procure une solution réalisable pour le problème

maître. Une solution duale est associée à chaque solution réalisable. La résolution du sous-problème permet de déterminer la variable possèdant le plus petit coût réduit. À l'itération i, le coût réduit $\bar{c}_p^i$ d'une variable θ_p telle que $b_p^k = 1$ se calcule comme suit : $\bar{c}_p^i = c_p - \sum_{j \in J} a_{jp}\pi_j^i - \sigma_k^i$, où

c_p : valeur du coefficient de la variable θ_p dans la fonction objectif (1.6),

a_{jp} : valeurs des coefficients de la variable θ_p dans les contraintes (1.7),

π_j^i : valeurs, à l'itération i, des variables duales associées aux contraintes (1.7),

σ_k^i : valeurs, à l'itération i, des variables duales associées aux contraintes (1.8).

C'est donc dire que le coût réduit d'une variable change à chaque itération puisqu'il varie en fonction des variables duales générées par la résolution du problème maître restreint. La méthode de génération de colonnes se termine quand le sous-problème ne génère plus de variables ayant un coût réduit négatif. En d'autres termes, la méthode de génération de colonnes s'achève si, lors de la résolution du sous-problème, il est impossible de trouver une variable qui peut contribuer, par son coût réduit négatif, à faire diminuer la valeur de la fonction objectif du problème maître restreint. Par contre, si une telle variable existe, il importe de l'ajouter au problème maître restreint et de résoudre ce dernier lors de l'itération suivante.

Étant donné le modèle de partitionnement d'ensemble pour le PTVDM, le problème maître restreint consiste à résoudre le modèle (1.6)-(1.9) limité à un sous-ensemble de variables, alors que le sous-problème se résout en appliquant un algorithme de plus court chemin. Il importe de souligner que la méthode de génération de colonnes est plus efficace lorsque le sous-problème est facile à résoudre. C'est d'ailleurs le cas ici puisque les algorithmes de résolution pour les problèmes de plus court chemin sont polynomiaux.

1.3.2 L'agrégation dynamique de contraintes

Comme il en a été question dans la sous-section précédente, la génération de colonnes est une technique utilisée pour résoudre des problèmes comportant un très grand nombre de variables. Cependant, il arrive que cette technique devienne inefficace. En fait, la présence de nombreuses contraintes de type partitionnement d'ensemble et/ou de plusieurs colonnes contenant de nombreux éléments non-nuls (plus de dix) entraîne une augmentation du temps de calcul du problème maître due à la dégénérescence. À ce moment-là, la méthode d'agrégation dynamique de contraintes devient intéressante puisqu'elle consiste à réduire le nombre de contraintes de type partitionnement d'ensemble en regroupant certaines d'entre elles selon une *relation d'équivalence*, notée R^C. ELHALLAOUI ET AL. (2003) définissent la relation d'équivalence comme suit :

> En considérant un ensemble de tournées C, deux tâches w_1 et $w_2 \in W$ sont dites *équivalentes* en regard de C si chacun des chemins compris dans C couvre à la fois les tâches w_1 et w_2 ou aucune d'entre elles.

Cette relation d'équivalence permet de répartir les tâches en des *classes d'équivalence* distinctes. Afin de mieux comprendre le concept de classe d'équivalence, il importe d'introduire la notation suivante :

$$L : \text{l'ensemble des classes d'équivalence,}$$
$$W_l : \text{le sous-ensemble de tâches dans la classe } l \in L,$$
$$Q = \{W_l : l \in L\} : \text{la partition des tâches en sous-ensembles } W_l.$$

À partir de ces définitions, ELHALLAOUI ET AL. (2003) énoncent un critère permettant d'identifier si une tournée p est compatible avec une classe d'équivalence $l \in L$. Le *critère de compatibilité* se définit comme suit :

> En considérant une tournée p comprise dans P^k, $k \in K$, et T_p l'ensemble des tâches couvertes par cette tournée, il est possible de dire que la tournée

p est *compatible* avec la classe d'équivalence $l \in L$ si $W_l \cap T_p$ est vide ou égale à W_l.

Aussi, la tournée p est dite compatible avec la partition Q si p est compatible avec toutes les classes d'équivalence comprises dans L. Dans ce cas, θ_p^k ou sa colonne correspondante est dite compatible avec Q.

L'agrégation dynamique de contraintes, au lieu d'utiliser le problème maître restreint (PMR) traditionnellement associé à la génération de colonnes, se sert d'un problème maître restreint agrégé ($PMRA$). Ce dernier considère un sous-ensemble de variables et de contraintes plus petit que le PMR.

L'agrégation dynamique de contraintes est un processus itératif qui peut être décrit comme suit. Dans un premier temps, une partition Q est définie à partir d'une heuristique ou d'une solution initiale. Cette partition correspond à l'ensemble des tâches couvertes par les tournées comprises dans la solution initiale (ou heuristique). Tout au long du processus de résolution, Q sera modifiée dynamiquement par le changement de la composition de C, et ce jusqu'à ce que la solution optimale soit trouvée. Deuxièmement, le $PMRA$ est résolu en considérant la partition Q. Il en résulte une solution primale x de valeur Z et une solution duale agrégée $\hat{\alpha}$ associée à la solution primale. À ce point, il faut désagréger la solution duale $\hat{\alpha}$ en appliquant un algorithme de plus court chemin. La solution duale désagrégée est notée α et elle permet d'obtenir un ensemble de colonnes de coût réduit négatif. Advenant le cas où cet ensemble serait vide, alors la solution courante s'avère optimale pour le problème complet et l'algorithme de résolution s'arrête. Par contre, si l'ensemble contient des colonnes de coût réduit négatif, alors celles-ci doivent être ajoutées aux colonnes déjà comprises dans le $PMRA$. Pour ce faire, il faut que l'algorithme de résolution décide s'il est préférable de modifier la partition Q actuelle (itération majeure) ou de la conserver (itération mineure). Si les colonnes choisies sont compatibles avec la partition Q, alors

elles sont ajoutées aux autres colonnes déjà présentes dans le $PMRA$. Le $PMRA$ est résolu à nouveau et, une fois de plus, une solution primale x de valeur Z et une solution duale agrégée $\hat{\alpha}$ associée à la solution primale sont obtenues. Cela constitue une itération mineure. À l'opposé, si les colonnes choisies sont incompatibles avec la partition Q, alors l'algorithme de résolution peut décider de les ajouter quand même au $PMRA$. En d'autres termes, la partition Q est brisée et une nouvelle partition Q' est créée. Par la suite, l'algorithme de résolution génère une solution primale x de valeur Z et une solution duale agrégée $\hat{\alpha}$ associée à la solution primale. Cela constitue une itération majeure parce que la partition Q a été brisée et reconstruite. L'algorithme de résolution continue ainsi jusqu'à ce que la solution courante du $PMRA$ soit optimale pour le problème complet, c'est-à-dire jusqu'à ce qu'aucune colonne de coût réduit négatif ne soit trouvée.

ELHALLAOUI ET AL. (2003) ont appliqué le principe d'agrégation dynamique de contraintes au problème de planification simultanée du personnel et des véhicules en transport urbain ; un problème constitué d'un grand nombre de contraintes de type partitionnement d'ensemble. Les résultats rapportés par ELHALLAOUI ET AL. (2003) montrent bien l'efficacité de la méthode d'agrégation dynamique de contraintes à résoudre des problèmes de grande taille. En fait, l'agrégation dynamique de contraintes a permis de diminuer le temps de résolution du problème maître par un facteur de 8, en augmentant de façon négligeable le temps de résolution des sous-problèmes. Dans le cadre de ce projet, nous utiliserons l'agrégation dynamique de contraintes pour produire les résultats présentés au chapitre 3.

1.3.3 La description des instances aléatoires

Toutes les instances utilisées pour effectuer les tests présentés dans ce mémoire ont été générées aléatoirement selon le modèle présenté dans CARPANETO ET AL. (1989).

Ces instances permettent de simuler la réalité du transport urbain par autobus. Les *points de relève* $\rho_1, \rho_2, ..., \rho_v$ se définissent comme étant les lieux physiques où les parcours débutent ou terminent. Les points de relève sont déterminés aléatoirement à l'intérieur d'une matrice carrée (60 × 60). Conséquemment, les distances à parcourir entre deux points de relève sont calculées selon la distance euclidienne $\theta_{a,b}$, où a et b sont respectivement les points de relève initial et final. Pour ce qui est de la génération des parcours, des points de relève initial ($p_i^{'}$) et final ($p_i^{''}$) sont générés pour chaque parcours T_i. Ainsi, la distance associée aux *arcs interparcours*, appellés aussi *arcs intertâches*, est représentée par $d_{i,j}$ qui est égal à $\theta_{p_i^{''},p_j^{'}}$. L'heure de départ ($s_i$) et l'heure de fin ($e_i$) de la tâche T_i sont générées selon deux types de parcours : les *parcours courts* et les *parcours longs.*

(a) Les parcours courts ont une probabilité d'occurrence de 40%. Leur heure de départ est choisie aléatoirement dans les intervalles [420,480] avec une probabilité de 15%, [480,1020] avec une probabilité de 70% et [1020,1080] avec une probabilité de 15%. Leur heure de fin est distribuée aléatoirement dans l'intervalle [s_i+$d_{p_i^{'},p_i^{''}}$+5, $s_i + d_{p_i^{'},p_i^{''}}$+40].

(b) Les parcours longs ont une probabilité d'occurrence de 60%. Leur heure de départ est choisie aléatoirement dans l'intervalle [300,1200]. Leur heure de fin est distribuée aléatoirement dans l'intervalle [s_i+180,s_i+300]. Enfin, il importe de noter que les points de relève initial et final sont les mêmes pour tous les parcours longs, i.e. $p_i^{'} = p_i^{''}$.

La localisation des dépôts est définie selon la classe du problème à résoudre. Si la *classe A* est choisie, alors les dépôts sont disposés aléatoirement à l'intérieur de la matrice carrée (60 × 60). Si la *classe B* est choisie, alors la localisation des dépôts est assujettie aux règles suivantes :

- 2 dépôts : Les dépôts sont placés dans les coins opposés de la matrice carrée (60 × 60) ;
- 3 dépôts : Deux dépôts sont placés dans les coins opposés de la matrice carrée (60 × 60) et la localisation du troisième dépôt est choisie aléatoirement ;

- 4 dépôts : Les dépôts sont placés dans les quatre coins de la matrice carrée (60 × 60) ;
- plus de 4 dépôts : Quatre dépôts sont placés dans les quatre coins de la matrice carrée (60 × 60) et la localisation des autres dépôts est choisie aléatoirement.

Le nombre r_k de véhicules disponibles dans chaque dépôt D_k est choisi aléatoirement dans l'intervalle $[3 + n/3m, 3 + n/2m]$. Cet intervalle assure la faisabilité des instances à résoudre.

Les coûts c_{ij} associés aux arcs se définissent comme suit :

- $c_{ij} = \lfloor 10d_{ij} + 2(s_j - e_i - d_{ij}) \rfloor$, pour tous les parcours compatibles (T_i, T_j) ;
- $c_{kj} = \lfloor 10$(distance euclidienne entre le dépôt et le lieu où le parcours débute)$\rfloor + 5000$, pour tous les dépôts D_k et les parcours T_j ;
- $c_{ik} = \lfloor 10$(distance euclidienne entre le lieu où le parcours termine et le dépôt)$\rfloor + 5000$, pour tous les parcours T_i et les dépôts D_k.

Ainsi, l'utilisation d'un véhicule supplémentaire dans la solution optimale est pénalisée. La fonction objectif permet donc de minimiser à la fois les coûts opérationnels de transport et les coûts liés à l'utilisation d'un véhicule supplémentaire dans la solution optimale.

CHAPITRE 2 : LA MÉTHODE DE HADJAR ET AL. MODIFIÉE

La première partie de ce chapitre porte sur une étude approfondie de la méthode proposée dans HADJAR ET AL. (2003) pour résoudre le PTVDM. Dans le cadre de cette étude, une série de tests a été produite pour étudier le comportement de cette méthode par rapport à différents facteurs. À la lumière de l'information fournie par ces tests, une nouvelle méthode de branchement a été développée et appliquée à la partie heuristique de l'arbre de branchement décrit dans HADJAR ET AL. (2003). Une description complète de cette méthode de branchement, ainsi que des résultats sont présentés dans la section 2.2.

Dans ce mémoire, un type de problème se caractérise par la taille de ses instances ou, en d'autres termes, par le nombre de tâches et de dépôts qui le composent. Par exemple, la notation utilisée pour désigner les problèmes comptant 300 tâches et quatre dépôts est « 300t-4d ». Tous les problèmes sont de classe B (voir 1.3.3). Tous les tests de cette section ont été réalisés avec la version 4.2 de GENCOL, un logiciel développé par une équipe de recherche montréalaise du GERAD. Tous les tests ont été effectués avec un ordinateur SUN ULTRA10 de 440Mhz.

2.1 L'étude de la méthode de Hadjar et al. (2003)

Les résultats présentés dans HADJAR ET AL. (2003) montrent que le temps de résolution de la partie exacte de l'arbre de branchement (sans plan coupant) tend à augmenter avec le nombre de tâches et de dépôts contenus dans l'instance à solutionner. Par exemple, en considérant des groupes de dix instances aléatoires de type

B, construites selon le modèle énoncé dans CARPANETO ET AL. (1989), et comptant six dépôts et respectivement 200, 400, 600 et 800 tâches, il est possible de constater que le temps moyen de résolution de la partie exacte de l'arbre de branchement passe rapidement de 38,9 à 2327,1 à 57576 à 180000 secondes.

Conséquemment, il est pertinent de s'interroger sur la relation entre la qualité de la solution entière trouvée dans la partie heuristique, le pourcentage de variables éliminées au noeud 0 et le temps de résolution de la partie exacte de l'arbre de branchement. HADJAR ET AL. (2003) proposent d'éliminer toutes les variables ayant un coût réduit supérieur ou égal à un certain *seuil*, qu'ils définissent comme étant la différence entre la valeur de la solution entière trouvée dans la partie heuristique (Z_{IP}) et la valeur de la solution duale de la relaxation linéaire obtenue au noeud 0 (Z_{LP}). Ces variables sont éliminées puisqu'elles ne peuvent faire partie de la solution optimale entière. Étant donné que le principe d'élimination des variables s'applique au noeud racine de l'arbre de branchement, c'est-à-dire juste avant de débuter l'exploration de la partie exacte, il est raisonnable de penser que, plus la proportion de variables fixées à 0 (ou éliminées du problème) est élevée, plus le temps de résolution de la partie exacte de l'arbre de branchement peut être réduit. Afin d'étudier les effets de la qualité de la solution heuristique, nous allons considérer des *seuils artificiels* et analyser leurs impacts sur le pourcentage de variables éliminées et le temps de résolution de la partie exacte de l'arbre de branchement.

Le premier test consiste donc à vérifier l'impact en termes de pourcentage de variables éliminées d'un seuil artificiel de plus en plus petit. Pour ce test, dix instances de chaque type de problème ont été résolues. La moyenne de ces dix instances est présentée dans le tableau 2.1.

Tableau 2.1 – Pourcentage de variables éliminées en fonction du seuil artificiel

Variables éliminées (en pourcentage)						
Seuil artificiel	**300t-4d**	**400t-4d**	**500t-4d**	**300t-6d**	**400t-6d**	**500t-6d**
10	98	99	99	98	99	99
30	98	98	98	98	98	98
50	97	97	98	97	98	98
100	94	95	95	95	96	96
300	80	83	80	80	84	84
500	65	68	63	66	70	69

En observant le tableau 2.1, il est possible de voir que, de manière générale, plus le seuil artificiel est petit, plus il est possible de fixer des variables à 0. Aussi, lorsque le seuil artificiel augmente, le pourcentage de variables éliminées a tendance à diminuer ; ce qui va dans le même sens que notre hypothèse initiale. De plus, il est intéressant de constater que le pourcentage de variables éliminées reste assez semblable pour un même seuil artificiel, et ce même si la taille des instances croît. Ces constatations sont à la base même de cette recherche puisqu'elles nous incitent à trouver une solution heuristique de la meilleure qualité possible dans le but de générer un seuil artificiel plus petit permettant d'éliminer davantage de variables au noeud 0. L'élimination d'une proportion élevée de variables pourrait, il faut le rappeler, permettre de gagner du temps lors de la résolution de la partie exacte de l'arbre de branchement.

Dans le même ordre d'idées, deux nouveaux critères pour éliminer les variables ont été testés. Ces critères sont testés pour prendre connaissance des gains potentiels en termes de temps total de résolution, et ce lorsque le seuil artificiel est le plus petit possible. Ainsi, le tableau 2.2 est divisé verticalement en trois sections. La première section présente les résultats obtenus par la méthode de HADJAR ET AL. (2003) lorsque celle-ci est appliquée sans utiliser les plans coupants. Cette section, présentée

à titre comparatif, comprend des résultats concernant le pourcentage de variables éliminées (%éli.), le temps total de résolution (tt) comprenant le temps alloué à la résolution de la partie heuristique (th) et de la relaxation linéaire au noeud 0 (tRL). La deuxième section (Méth. du ÉI + ϵ) présente les résultats obtenus lorsque le critère utilisé pour attribuer une valeur nulle aux variables est le suivant : *toutes les variables dont le coût réduit est supérieur ou égal à l'*écart d'intégrité *(+ ϵ) doivent être éliminées du problème.* L'écart d'intégrité se définit comme étant la différence entre la valeur optimale de la relaxation linéaire au noeud 0 et la valeur optimale de la solution entière. L'écart d'intégrité constitue donc le meilleur critère qu'il est possible d'obtenir à partir de la borne supérieure. La troisième section (Méth. du CR + ϵ) présente les résultats obtenus lorsque le critère utilisé pour attribuer une valeur nulle aux variables est le suivant : *toutes les variables dont le coût réduit est supérieur ou égal au coût réduit maximal au noeud 0 des variables ayant une valeur égale à 1 dans la solution optimale entière (+ ϵ) doivent être éliminées du problème.* Ce critère permet de conserver toutes les variables faisant partie de la solution optimale entière. Il s'agit donc du critère le plus serré parmi les deux proposés ici lorsque ϵ tend vers 0. La valeur de ϵ a été fixée à 0,5 pour ces deux tests afin d'éviter les erreurs d'arrondis. Pour chacun de ces deux tests, les résultats observés concernent le pourcentage de variables éliminées (%éli.) et le temps total de résolution (tt) comprenant le temps consacré à la relaxation linéaire au noeud 0. Pour chacun des deux critères testés, un pourcentage d'amélioration (%am) du temps de résolution de la partie exacte de l'arbre de branchement est calculé par rapport à la méthode de référence. Étant donné que le pourcentage d'amélioration concerne seulement la partie exacte de l'arbre de branchement, il importe de soustraire au temps total (tt) le temps consacré à la résolution de la relaxation linéaire (tRL) (pour les trois méthodes) et celui consacré à la résolution de la partie heuristique (th) (pour la méthode de HADJAR ET AL. (2003) seulement). Les résultats présentés dans le tableau 2.2 sont des moyennes calculées sur cinq instances du même type de problème. Tous les temps dans le tableau 2.2 sont exprimés en secondes.

Tableau 2.2 – Tableau comparatif des différentes méthodes

	Hadjar et al.(2003)				Méth. du ÉI + ϵ			Méth. du CR + ϵ		
taille	**%éli.**	**tt(s)**	**th(s)**	**tRL(s)**	**%éli.**	**tt(s)**	**%am**	**%éli.**	**tt(s)**	**%am**
300t-4d	93	142	31	64	97	92	40	98	83	60
400t-4d	91	955	89	182	97	907	-6	98	450	61
500t-4d	81	4362	210	412	97	4580	-11	99	1452	72
300t-6d	91	243	53	62	97	140	39	98	86	81
400t-6d	86	1246	140	160	97	907	21	99	366	78
500t-6d	84	22583	299	305	96	21524	3	99	13556	40

Les résultats du tableau 2.2 montrent que les deux critères testés permettent de fixer à 0 entre 96% et 99% du nombre total de variables. La méthode se basant sur le coût réduit maximal affiche des pourcentages d'amélioration du temps de résolution de la partie exacte de l'arbre de branchement supérieurs à ceux produits par la méthode s'appuyant sur l'écart d'intégrité. Cela est principalement dû au fait que le coût réduit maximal est le critère le plus précis parmi les deux critères proposés lorsque ϵ tend vers 0. Dans le cas des problèmes composés de 400 tâches et de quatre dépôts (400t-4d) et de 500 tâches et de quatre dépôts (500t-4d), le pourcentage d'amélioration est négatif pour la méthode se basant sur l'écart d'intégrité. Cela signifie que cette méthode a pris, en moyenne, plus de temps à résoudre ces deux types de problèmes que la méthode de HADJAR ET AL. (2003).

Les résultats consignés dans cette première section supportent donc l'hypothèse voulant qu'un écart artificiel petit permette d'éliminer une grande proportion de variables au noeud racine de l'arbre de branchement. Conséquemment, cela permet de réduire le temps de résolution de la partie exacte de l'arbre de branchement.

2.2 La nouvelle méthode heuristique

À la lumière de l'information recueillie lors des deux premiers tests, une nouvelle avenue est explorée afin de définir une méthode heuristique basée sur un principe de «cluster-first, route-second ». Cette avenue consiste à vérifier le nombre de dépôts couvrant chaque tâche dans la solution de la relaxation linéaire au noeud 0. Le tableau 2.3 présente les résultats pour cinq instances de chaque type de problème. Une ligne du tableau 2.3 se lit de la façon suivante : «Pour la première instance comptant 300 tâches et quatre dépôts (300t-4d(a)), dans la solution de la relaxation linéaire, 200 tâches sont couvertes par un seul dépôt, 81 par deux dépôts, 18 par trois dépôts et une seule tâche est couverte par quatre dépôts. »

Les résultats présentés dans le tableau 2.3 permettent de constater, qu'en moyenne, le 2/3 des tâches est affecté à un seul dépôt dans la solution de la relaxation linéaire au noeud 0. Par contre, cela ne signifie pas pour autant que les tâches sont couvertes par un seul véhicule. En fait, il peut arriver que plusieurs véhicules couvrent des fractions de tâches. Par contre, si tous les véhicules couvrant une tâche proviennent du même dépôt, alors il est possible de conclure qu'un seul dépôt couvre cette tâche.

Ces résultats montrent, qu'en se basant sur la solution de la relaxation linéaire du noeud 0, il est possible d'obtenir rapidement une solution réalisable. En fait, les tâches sont d'abord affectées aux dépôts (principe de « cluster-first »). Par la suite, une tournée est définie pour chaque véhicule dans chaque dépôt (principe de « route-second »).

Tableau 2.3 – Nombre de dépôts affectés par tâche dans la solution de la relaxation linéaire au noeud 0

Nombre de dépôts affectés par tâche						
Nombre de dépôts	**1**	**2**	**3**	**4**	**5**	**6**
300t-4d(a)	200	81	18	1	-	-
300t-4d(b)	201	81	17	1	-	-
300t-4d(c)	223	65	11	1	-	-
300t-4d(d)	228	67	5	0	-	-
300t-4d(e)	223	63	14	0	-	-
400t-4d(a)	281	96	17	6	-	-
400t-4d(b)	283	93	24	0	-	-
400t-4d(c)	251	121	28	0	-	-
400t-4d(d)	298	75	23	4	-	-
400t-4d(e)	296	85	19	0	-	-
500t-4d(a)	372	112	14	2	-	-
500t-4d(b)	358	106	36	0	-	-
500t-4d(c)	353	117	30	0	-	-
500t-4d(d)	351	114	34	1	-	-
500t-4d(e)	366	112	20	2	-	-
300t-6d(a)	204	62	30	4	0	0
300t-6d(b)	201	77	19	3	0	0
300t-6d(c)	193	94	13	0	0	0
300t-6d(d)	195	88	15	2	0	0
300t-6d(e)	227	57	16	0	0	0
400t-6d(a)	263	102	33	2	0	0
400t-6d(b)	246	112	40	2	0	0
400t-6d(c)	268	110	18	4	0	0
400t-6d(d)	259	103	32	5	1	0
400t-6d(e)	253	114	28	5	0	0
500t-6d(a)	332	118	46	4	0	0
500t-6d(b)	315	146	38	1	0	0
500t-6d(c)	367	109	24	0	0	0
500t-6d(d)	338	117	36	9	0	0
500t-6d(e)	320	140	33	7	0	0

À la suite de l'observation des résultats du tableau 2.3, une nouvelle heuristique a été élaborée. Celle-ci a pour but l'obtention d'une solution réalisable ; elle a donc le même rôle que la branche heuristique dans l'arbre de branchement décrit par HADJAR ET AL. (2003). Les lignes qui suivent expliquent les grandes idées de cette heuristique, appelée méthode d'« affectation tâche-dépôt », qui est intégrée dans un arbre de branchement exploré selon une stratégie de type *profondeur d'abord*. Ce type de stratégie d'exploration ne permet pas de retour en arrière. Aucune autre méthode de branchement n'est utilisée. Dès qu'une solution réalisable est trouvée, alors elle est utilisée pour procéder à l'élimination des variables au noeud 0 de l'arbre de branchement, c'est-à-dire avant de débuter l'exploration de la partie exacte. D'ailleurs, la résolution de la partie exacte suit les mêmes règles que celles décrites à la sous-section 1.2.3.

À chacun des noeuds de l'arbre de branchement, toutes les tâches sont placées, par ordre décroissant, selon le flot total passant sur chaque tâche et provenant de chaque dépôt. Comme il en a été question au début de cette section, une tâche peut être couverte par deux véhicules provenant du même dépôt. Par exemple, elle peut être couverte à raison de 25% par le premier véhicule et de 75% par le second. Dans ce cas, l'algorithme détecte que la tâche est couverte à 100% par le dépôt abritant ces deux véhicules. Le flot total passant sur cette tâche et provenant de ce dépôt est donc égal à un. Par la suite, les tâches sont examinées en respectant l'ordre établi par la liste. Les tâches sont traitées de façon distincte selon le flot passant sur chacune. Trois cas de figure sont possibles :

(i) Le flot passant sur la tâche est égal à 1. Le nombre de tâches dont le flot est égal à 1 est noté *Nbflot1*.

(ii) Le flot passant sur la tâche est fractionnaire, mais supérieur ou égal à un seuil de tolérance préalablement défini. Le nombre de tâches dont le flot est fractionnaire, mais supérieur ou égal à ce seuil de tolérance est noté *NbSeuil*.

(iii) Le flot passant sur la tâche est strictement inférieur à un seuil de tolérance préalablement défini.

À chaque noeud de l'arbre de branchement, trois règles doivent être respectées :

(i) Toutes les tâches dont le flot provenant d'un dépôt est strictement égal à 1 sont affectées à ce dépôt.

(ii) Un nombre minimal de tâches dont le flot est fractionnaire sont affectées à un dépôt. Ce nombre est noté *NbMin*.

(iii) Un nombre maximal de tâches sont affectées à un dépôt. Ce nombre, noté *NbMax*, est calculé en fonction du nombre de tâches pas encore affectées et du pourcentage maximal de tâches devant être affectées à chaque noeud de l'arbre de branchement.

En respectant ces règles, le nombre de tâches affectées à un dépôt à chaque noeud de l'arbre de branchement est donc égal à :

$$Max\{Nbflot1 + NbMin, Min\{Nbflot1 + NbSeuil, NbMax\}\}$$

lorsque l'on veut en maximiser la valeur. Dans cette expression, le minimum découle de la règle (iii) et le maximum découle de la règle (ii).

La nouvelle méthode heuristique permet de partitionner l'ensemble des tâches parmi les dépôts. Une fois cette première étape accomplie, il ne reste plus qu'à résoudre un problème de flot à coût minimum pour chaque dépôt pour obtenir une solution réalisable. Cependant, comme seule la valeur de la fonction objectif de cette solution est nécessaire pour procéder à l'élimination des variables, il n'est pas obligatoire de résoudre les problèmes de flot à coût minimum puisque la valeur optimale de la solution de leur relaxation linéaire est égale à la valeur optimale de leur solution en nombres entiers. Il s'agit donc de résoudre simplement la relaxation linéaire pour obtenir Z_{IP}.

Le but directement recherché par la modification de la méthode de HADJAR ET AL. (2003) est l'obtention d'une solution réalisable de meilleure qualité. Comme il en a été question à quelques reprises dans ce mémoire, plus la solution obtenue dans la partie heuristique est près de la solution optimale du problème à résoudre, plus le temps de résolution de la partie exacte de l'arbre de branchement peut être réduit puisqu'une solution heuristique d'une grande précision permet l'élimination d'une proportion élevée de variables. Ainsi, il est possible de dire que le but indirectement recherché par la modification de la méthode de HADJAR ET AL. (2003) est l'obtention d'un temps de résolution réduit pour la partie exacte de l'arbre de branchement. Il est aussi permis d'espérer que la nouvelle méthode heuristique permettra de diminuer le temps de résolution de la partie heuristique, sans toutefois altérer la qualité de la solution obtenue.

Une analyse de sensibilité a été effectuée sur les plus gros problèmes étudiés depuis le début de ce mémoire. Cette analyse avait pour but d'observer l'impact sur le pourcentage de variables éliminées et le temps total de résolution (parties heuristique et exacte confondues) de la variation de deux paramètres (pourcentage maximal de tâches devant être fixées à chaque noeud par rapport au nombre de tâches non encore affectées et seuil de tolérance permettant d'affecter une tâche à un dépôt). Dans le but de produire une solution heuristique de la meilleure qualité possible, le seuil minimal de tâches devant être affectées et dont le flot est fractionnaire est fixé à 1 (sans possibilité de changement). Le tableau 2.4 présente les différentes configurations de paramètres testées, tandis que le tableau 2.5 montre la performance de chacune de ces configurations en termes de pourcentage de variables éliminées et de temps total de résolution (parties heuristique et exacte confondues) pour quatre instances de quatre types de problèmes différents, c'est-à-dire 400t-4d, 400t-6d, 500t-4d et 500t-6d.

Tableau 2.4 – Description des paramètres selon les différentes configurations testées

	Paramètres		
Configurations	**Nb min. à choisir**	**% max. à choisir**	**Seuil de tolérance**
A	1	50	0,7
B	1	75	0,7
C	1	50	0,8
D	1	75	0,8
E	1	50	0,6
F	1	65	0,7
G	1	40	0,7
H	1	60	0,7

Dans le tableau 2.5, les temps totaux de résolution ont été placés par ordre croissant, en débutant par la configuration ayant été la plus rapide à résoudre l'instance et en terminant par la configuration ayant été la plus lente. Pour ce qui est des pourcentages de variables éliminées, ils ont plutôt été placés en ordre décroissant. En fait, selon notre hypothèse initiale, plus le pourcentage de variables éliminées est grand, plus la configuration a bien performé. Pour des fins de comparaison, la performance de la méthode de référence (HADJAR ET AL. (2003)), désignée par Had dans le tableau 2.5, a été ajoutée aux classements des configurations.

Tableau 2.5 – Classement des configurations selon leur temps total de résolution et leur pourcentage de variables globalement éliminées

400t-4d				400t-6d			
tt(s)	**Config.**	**%éli.**	**Config.**	**tt(s)**	**Config.**	**%éli.**	**Config.**
1437	Had	94,7	A	1037	H	92,1	B
1558	A	94,7	F	1045	A	86,1	Had
1630	H	94,7	H	1049	F	85,9	A
1644	D	93,4	Had	1434	G	85,9	E
1674	F	92,2	G	1504	E	85,9	F
1755	B	89,8	B	1744	Had	85,9	G
1937	C	85,6	D	1760	B	85,9	H
2019	G	75,6	E	1897	C	79,8	D
2051	E	58,2	C	2045	D	71,6	C
500t-4d				500t-6d			
tt(s)	**Config.**	**%éli.**	**Config.**	**tt(s)**	**Config.**	**%éli.**	**Config.**
2093	A	96,6	Had	34038	G	87	D
2597	H	95,3	E	35139	Had	76,8	B
2621	B	90,1	A	35531	A	75,9	A
2988	D	90,1	F	35672	E	75,9	E
3325	G	90,1	G	40829	D	74,8	G
3455	Had	89,1	C	41334	C	72,2	Had
3765	F	88,6	D	51974	B	65,5	C
3908	E	84,9	B	>25h	F	N/A	F
3934	C	79,4	H	>25h	H	N/A	H

En comparant la performance globale des huit configurations, nous avons choisi de fixer les paramètres comme dans la configuration A. En fait, cette dernière procure des résultats assez respectables puisqu'elle se classe toujours dans les trois premiers rangs autant en termes de temps total de résolution que de pourcentage de variables éliminées, et ce pour toutes les tailles d'instances testées dans le cadre de l'analyse de sensibilité. Ainsi, le seuil de tolérance permettant d'affecter une tâche à un dépôt est fixé à 0,7 et le pourcentage maximal de tâches devant être fixées à chaque noeud

par rapport au nombre de tâches non encore affectées est établi à 50%.

Lors de l'analyse de sensibilité, il a été possible de constater que certains problèmes prenaient beaucoup de temps à achever la résolution de la partie heuristique. En fait, à cause de la dégénérescence, l'algorithme primal du simplexe pouvait itérer plusieurs milliers de fois à l'intérieur d'un même noeud ; ce qui contribuait à augmenter considérablement les temps de calcul. Ainsi, il a fallu permettre l'utilisation de l'algorithme dual du simplexe. La règle suivante a donc été ajoutée à la méthode de branchement proposée précédemment : pour un même noeud, si la valeur de la fonction objectif n'a pas diminué de plus d'une valeur ϵ égale à 1×10^{-7} après 25 itérations de l'algorithme primal du simplexe, alors la résolution du présent noeud doit se terminer en utilisant l'algorithme dual du simplexe.

Le tableau 2.6 présente une comparaison entre la méthode de référence (HADJAR ET AL. (2003)) et la méthode dite d'affectation tâche-dépôt. Pour chacune de ces deux méthodes, le pourcentage de variables éliminées (%éli.), le temps total de résolution (tt) incluant le temps alloué à la résolution de la partie heuristique (th) et de la relaxation linéaire au noeud 0 (tRL), et le nombre de noeuds (no) explorés au total (parties exacte et heuristique confondues) sont présentés. Aussi, pour comparer la performance de la nouvelle méthode par rapport à la méthode de HADJAR ET AL. (2003), trois pourcentages d'amélioration sont donnés. Les pourcentages d'amélioration am1, am2 et am3 concernent respectivement le temps total de résolution, le temps de résolution de la partie heuristique et le pourcentage de variables éliminées. Ici, il faut apporter une précision concernant le signe du pourcentage d'amélioration am3. En fait, lorsque le pourcentage de variables éliminées est plus grand dans la méthode d'affectation tâche-dépôt que dans la méthode de HADJAR ET AL. (2003), cela signifie que la nouvelle méthode a éliminé plus de variables que la méthode de référence. Cependant, dans ce cas, le pourcentage d'amélioration am3 est négatif. Conséquemment, lorsque le signe de ce dernier est positif, cela signifie qu'il y a eu

Tableau 2.6 – Tableau comparatif entre la méthode de HADJAR ET AL. (2003) et la méthode d'affectation tâche-dépôt

	Hadjar et al.(2003)					Méth. d'affectation tâche-dépôt						
taille	**tRL(s)**	**tt(s)**	**th(s)**	**%éli.**	**no**	**tt(s)**	**am1(%)**	**th(s)**	**am2(%)**	**%éli.**	**am3(%)**	**no**
300t-4d	64	142	31	93	78	139	2	24	23	90	3	82
400t-4d	182	955	89	91	521	1097	-15	76	14	88	4	686
500t-4d	412	4362	210	81	1800	3944	10	126	40	88	-9	1760
300t-6d	62	243	53	91	231	206	15	29	46	85	6	188
400t-6d	160	1246	140	86	770	1046	16	74	47	83	3	687
500t-6d	305	22583	299	84	8900	29056	-29	169	44	86	-3	11373

une diminution du pourcentage de variables éliminées entre la méthode de référence et la nouvelle méthode. Les résultats présentés dans le tableau 2.6 sont en fait des moyennes calculées sur cinq instances de même taille. Tous les temps sont exprimés en secondes.

Le tableau 2.6 présente des résultats pour le moins intéressant en ce qui concerne le pourcentage d'amélioration du temps de résolution de la partie heuristique (am2). En fait, tous les types de problèmes montrent une amélioration, cette dernière variant entre 14% et 47%. Cela signifie que l'utilisation de la nouvelle méthode de branchement a permis d'accélérer la résolution de la partie heuristique permettant, du même coup, d'obtenir plus rapidement une solution réalisable pour procéder à l'élimination des variables au noeud 0 de l'arbre de branchement. Par contre, pour ce qui est de l'amélioration du temps total comprenant le temps de résolution consacré à la relaxation linéaire, à la partie heuristique et à la partie exacte de l'arbre de branchement, les pourcentages d'amélioration am1 sont plus variables se situant entre -29% et 16%. De plus, il n'est pas possible d'établir un lien certain entre la complexité des problèmes à résoudre et la détérioration des pourcentages d'amélioration am1. En fait, les problèmes comptant 500 tâches et quatre dépôts affichent une amélioration du temps total de résolution de 10%, alors que les problèmes comptant 400 tâches et quatre dépôts, donc d'une complexité moindre, montrent une détérioration du temps

total de résolution de 15%. Comme il est possible de le constater en observant le tableau 2.6, les instances comptant 500 tâches et quatre dépôts ont éliminées 9% plus de variables en utilisant la nouvelle méthode heuristique. Dans ce cas, le pourcentage d'amélioration du temps total (am1) est de 10%. Cela est en accord avec notre hypothèse initiale voulant que plus le pourcentage de variables éliminées est élevé, plus le temps total nécessaire pour résoudre la partie exacte est court. Pour ce qui est du second type de problème présentant une amélioration quant au pourcentage de variables éliminées (500t-6d), celui-ci montre une détérioration de 29% de son temps total de résolution (am1). À la lumière de ces résultats, il est contradictoire de conclure qu'une amélioration du pourcentage de variables éliminées (am3) entraîne nécessairement une diminution du temps total de résolution. Une fois de plus le compromis entre la qualité de la solution obtenue par l'application de la nouvelle méthode de branchement dans la partie heuristique et l'accélération de la résolution, en termes de temps total, est mis en évidence par les résultats présentés dans le tableau 2.6. Enfin, de manière générale, la nouvelle méthode de branchement proposée ne diminue pas nécessairement le nombre de noeuds de branchement explorés au total.

Pour remettre les chiffres du tableau 2.6 en perspective, le tableau 2.7 présente les résultats obtenus pour les cinq instances des deux types de problèmes suivants : 500 tâches et quatre dépôts et 500 tâches et six dépôts. Ce tableau revêt la même forme que le tableau 2.6. En observant les résultats contenus dans ce tableau, nous sommes à même de constater que, des dix instances présentées dans ce tableau, cinq (500t-4d(b), 500t-4d(d), 500t-4d(e), 500t-6d(c) et 500t-6d(d)) confirment notre hypothèse voulant que lorsque le pourcentage de variables éliminées est plus élevé en utilisant la nouvelle méthode, alors le pourcentage d'amélioration du temps total (am1) est positif, et inversement. Aussi, d'après les résultats du tableau 2.7, il ne semble pas y avoir de lien à établir entre la rapidité de la résolution de la partie heuristique et la qualité de la solution heuristique obtenue. En fait, pour toutes les instances

qui présentent une amélioration de leur pourcentage de variables éliminées (am3 négatif), il est également possible de noter une amélioration du temps heuristique (am2 positif).

Tableau 2.7 – Tableau comparatif entre la méthode de HADJAR ET AL. (2003) et la méthode d'affectation tâche-dépôt - instances individuelles

	Hadjar et al.(2003)					Méth. d'affectation tâche-dépôt						
taille	**tRL(s)**	**tt(s)**	**th(s)**	**%éli.**	**no**	**tt(s)**	**am1(%)**	**th(s)**	**am2(%)**	**%éli.**	**am3(%)**	**no**
500t-4d(a)	427	3455	116	97	1593	2774	20	126	-9	90	7	1215
500t-4d(b)	421	1319	315	84	126	705	47	88	72	96	-14	66
500t-4d(c)	400	9049	256	53	4056	11084	-22	223	13	76	-44	5273
500t-4d(d)	447	6635	195	75	2756	3788	43	114	42	90	-20	1801
500t-4d(e)	367	1352	169	97	471	1369	-1	76	55	91	7	445
500t-6d(a)	332	35139	311	72	13518	56012	-59	184	41	74	-3	21263
500t-6d(b)	298	6319	372	93	2731	5962	6	234	37	89	5	2799
500t-6d(c)	308	603	178	99	73	685	-14	68	62	96	3	143
500t-6d(d)	250	5548	269	84	2376	6005	-8	228	15	82	2	2419
500t-6d(e)	340	65306	366	71	25804	76617	-17	130	64	88	-24	30242

Le tableau 2.8 permet de constater que, sur 30 instances, la méthode d'affectation tâche-dépôt améliore 11 fois l'*erreur heuristique absolue* obtenue avec la méthode de HADJAR ET AL. (2003) (voir les nombres en caractères gras dans le tableau 2.8). L'*erreur heuristique absolue* se veut la différence entre la valeur de la solution réalisable trouvée par la méthode heuristique et la valeur de la solution optimale entière. Ces améliorations sont d'ailleurs considérables puisqu'elles varient entre 11,8% et 100% avec une moyenne de 54,2%. Dans le cadre de cette recherche, il est pertinent d'observer l'impact sur le temps total de résolution (parties heuristique et exacte confondues) d'une amélioration de l'erreur heuristique absolue entre les deux méthodes (voir colonne Am.1 dans le tableau 2.8). 6 fois sur 11 le temps total de résolution a diminué entre les deux méthodes (Am1 positif) par des pourcentages

variant entre 8,6% et 47,1% avec une moyenne de 33,7%. Pour les cinq autres instances présentant une amélioration de l'erreur heuristique absolue entre la méthode de HADJAR ET AL. (2003) et la méthode d'affectation tâche-dépôt, les pourcentages de détérioration du temps total de résolution (Am1 négatif) varient entre -59,4% et -10,6% avec une moyenne de -31,4%. Il n'est donc pas possible d'établir un lien entre l'amélioration de l'erreur heuristique absolue d'une instance et l'amélioration du temps total de résolution de cette même instance. Dans le cas des instances ayant subi une détérioration de leur erreur heuristique absolue, les pourcentages de détérioration varient entre 2% et 100% avec une moyenne de 52,5%. Les instances ayant subi une détérioration de leur erreur heuristique absolue n'ont pas nécessairement vu leur temps total de résolution augmenter par rapport à la méthode de HADJAR ET AL. (2003).

À la lumière des résultats présentés dans les tableaux 2.6, 2.7 et 2.8 et en guise d'épilogue à cette première partie sur les stratégies de résolution, il est possible de conclure que :

(a) La nouvelle méthode heuristique *affectation tâche-dépôt* permet d'accélérer considérablement la résolution de la partie heuristique.

(b) La qualité de la solution heuristique obtenue par la nouvelle méthode de branchement *affectation tâche-dépôt* est aussi bonne que celle obtenue, en plus de temps, par HADJAR ET AL. (2003).

(c) Au-delà d'un certain seuil, le fait d'éliminer plus de variables au noeud 0 de l'arbre de branchement n'entraîne pas nécessairement une diminution du temps total de résolution ; et inversement, c'est-à-dire le fait d'éliminer moins de variables au noeud 0 de l'arbre de branchement n'entraîne pas nécessairement une augmentation du temps total de résolution.

Il faut enfin préciser que ces conclusions reposent sur les résultats obtenus par la résolution des instances générées selon le modèle de CARPANETO ET AL. (1989). Ainsi, il est important d'appuyer sur le fait voulant que des instances générées par un autre modèle auraient pu donner des résultats quelque peu différents.

Tableau 2.8 – Pourcentage d'amélioration du temps total de résolution en relation avec l'erreur heuristique - instances individuelles

	Hadjar et al. (2003)	Méth. d'affectation tâche-dépôt	Am.1(%)
300t-4d(a)	46	49	-4,9
300t-4d(b)	20	378	-30
300t-4d(c)	116	124	-4,9
300t-4d(d)	8	84	2,9
300t-4d(e)	**141**	**40**	42,7
400t-4d(a)	**50**	**24**	-10,6
400t-4d(b)	**80**	**0**	8,6
400t-4d(c)	58	464	-18,7
400t-4d(d)	141	141	8,9
400t-4d(e)	**214**	**168**	-47,1
500t-4d(a)	10	133	19,7
500t-4d(b)	**295**	**65**	46,5
500t-4d(c)	**671**	**343**	-22,5
500t-4d(d)	**338**	**133**	42,9
500t-4d(e)	24	158	-1,3
300t-6d(a)	69	126	8,1
300t-6d(b)	**232**	**88**	47,1
300t-6d(c)	68	465	-9,3
300t-6d(d)	100	205	40,2
300t-6d(e)	25	25	-2,4
400t-6d(a)	147	150	41,8
400t-6d(b)	107	208	-12,7
400t-6d(c)	39	47	7,8
400t-6d(d)	**133**	**104**	14,1
400t-6d(e)	569	755	1,4
500t-6d(a)	**340**	**300**	-59,4
500t-6d(b)	114	219	5,7
500t-6d(c)	0	64	-13,6
500t-6d(d)	228	246	-8,2
500t-6d(e)	**417**	**132**	-17,3

CHAPITRE 3 : L'AGRÉGATION DYNAMIQUE DE CONTRAINTES APPLIQUÉE AU PTVDM

Ce chapitre présente une nouvelle méthode permettant de résoudre le problème de tournées de véhicules avec dépôts multiples (PTVDM). Il s'agit en fait d'une intégration de la technique d'agrégation dynamique de contraintes avec la méthode de résolution standard de HADJAR ET AL. (2003). Comme il a été mentionné dans la description de l'agrégation dynamique de contraintes (voir sous-section 1.3.2), cette technique requiert, pour enclencher la résolution, une partition initiale. C'est pourquoi ce chapitre est divisé en deux parties. Dans un premier temps, une méthode heuristique permettant de générer rapidement une solution réalisable pour le PTVDM est proposée. Par la suite, la nouvelle méthode de résolution est testée en utilisant comme données d'entrée une partition initiale découlant de la solution réalisable obtenue par la méthode heuristique expliquée à la section 3.1. Des résultats sont présentés et discutés à la fin de ce chapitre.

Tous les tests de cette section ont été réalisés avec la version 4.5 de GENCOL, un logiciel développé par une équipe de recherche montréalaise du GERAD. Tous les tests ont été effectués avec un ordinateur SUN ULTRA 10 de 440Mhz.

3.1 La recherche d'une solution réalisable

Comme il a été mentionné à la sous-section 1.2.1, BERTOSSI ET AL. (1987) ont été les premiers à montrer que le PTVDM est NP-complet pour $|K| \geq 2$. Par contre, lorsqu'il n'y a qu'un seul dépôt dans le problème, celui-ci peut être résolu en temps polynomial. En fait, le problème de tournées de véhicules à dépôt unique (PTVDU)

peut être résolu en utilisant un algorithme de flot à coût minimum. Le développement d'une méthode heuristique permettant de générer une solution réalisable au PTVDM s'appuie donc sur ces faits. Les lignes qui suivent expliquent le fonctionnement de cette heuristique fortement inspirée de celle présentée par OUKIL (2004).

La méthode heuristique se divise en trois étapes. Premièrement, le PTVDM est transformé en un PTVDU. Le réseau $G = (V, A)$, maintenant noté $G' = (V, E)$, est modifié selon les règles suivantes. D'abord, l'ensemble des $|K|$ noeuds représentant les dépôts est remplacé par un unique noeud, noté S, dont la capacité est égale à la somme des capacités des $|K|$ noeuds originaux. Le coût des arcs *dépôt-tâche* (resp. *tâche-dépôt*) est déterminé comme suit :

$$d_{S,j} = Min\{C_{n+k,j} : k \in K\},$$
$$d_{i,S} = Min\{C_{i,n+k} : k \in K\}.$$

Pour ce qui est des *arcs intertâches*, il faut se rappeler que, dans le PTVDM, toutes les paires de tâches compatibles (T_i, T_j) sont reliées par $|K|$ copies d'arcs intertâches. En fait, selon la définition du PTVDM spécifiée au début de ce mémoire, aucune contrainte de restriction tâche-dépôt n'est présente dans les problèmes à résoudre. Aussi, le seul facteur influençant le coût d'un arc intertâche est la distance entre le lieu où se termine la première tâche T_i et le lieu où débute la seconde tâche T_j. Ainsi, pour chaque copie d'un même arc intertâche, le coût est toujours identique dans le PTVDM. Conséquemment, dans le PTVDU, une seule copie des arcs intertâches est conservée. Aussi, son coût reste inchangé, mais il est maintenant noté d_{ij}.

Dans le modèle mathématique qui suit, une variable binaire Y_{ij} est associée à chaque arc $(i, j) \in E$. Y_{ij} est égale à 1 si un véhicule effectue successivement les tâches T_i et T_j, où i et $j \in \{1, 2, ..., n + 1\}$. Sinon, la variable Y_{ij} prend la valeur 0.

Le problème de tournées de véhicules à dépôt unique (PTVDU) s'écrit donc comme suit :

$$\text{Minimiser} \sum_{(i,j)\in E} d_{ij} Y_{ij} \tag{3.1}$$

sujet à

$$\sum_{j\in\delta^+(i)} Y_{ij} = 1 \qquad \forall i = 1, 2, ..., n \tag{3.2}$$

$$\sum_{j=1}^{n} Y_{S,j} \leq \sum_{k\in K} r_k \tag{3.3}$$

$$\sum_{j\in\delta^-(i)} Y_{ji} - \sum_{j\in\delta^+(i)} Y_{ij} = 0 \qquad \forall i \in N \tag{3.4}$$

$$Y_{ij} \in \{0, 1\} \qquad \forall (i, j) \in E. \tag{3.5}$$

Les contraintes (3.2) s'assurent que chaque tâche est couverte une et une seule fois. La contrainte (3.3) vérifie que le nombre de véhicules utilisés pour couvrir l'ensemble des tâches est inférieur ou égal au nombre total de véhicules disponibles au dépôt S, qui est lui-même égal à la somme des capacités des $|K|$ dépôts présents dans le PTVDM. Les contraintes (3.4) contrôlent la conservation de flot à tous les noeuds du graphe.

Une fois le PTVDM transformé en PTVDU, ce dernier est résolu à l'optimalité à l'aide de la version 9.0 du logiciel commercial CPLEX. Il s'agit de la seconde étape de la méthode heuristique. Troisièmement, les suites de tâches formant les tournées sont extraites de la solution optimale générée par l'optimiseur et sont réaffectées aux dépôts disponibles dans le PTVDM. En fait, pour chacune des tournées, la première et la dernière tâche, notées respectivement T_p et T_d, sont considérées. Puis, la somme des coûts c_{n+k,T_p} et $c_{T_d,n+k}$ des arcs *dépôt-*T_p et T_d*-dépôt* est calculée pour chaque

dépôt. De manière à ce que l'heuristique génère une solution réalisable la plus près possible de la solution optimale, il a été décidé que chaque tournée serait effectuée par un véhicule du dépôt présentant la plus petite somme si, évidemment, la capacité du dépôt en question le permet, c'est-à-dire si les véhicules de ce dépôt n'ont pas déjà tous été affectés à des tournées. Pour ce qui est des tournées comportant une seule tâche, le procédé est le même. En fait, dans ce cas, $T_p = T_d$, mais c_{n+k,T_p} n'est pas nécessairement égal à $c_{T_p,n+k}$ puisque le lieu de départ de la tâche T_p n'est pas nécessairement le même que son lieu d'arrivée. Toutefois, si c'est le cas, alors $c_{n+k,T_p} = c_{T_p,n+k}$.

Les trois étapes de la méthode heuristique se résume donc ainsi :

(i) Transformer le PTVDM en un PTVDU ;

(ii) Résoudre le PTVDU à l'optimalité avec un optimiseur commercial ;

(iii) Retirer de la solution optimale obtenue les tournées de véhicules et les affecter aux dépôts présents dans le PTVDM.

En croyant permettre l'accélération de la résolution de la relaxation linéaire au noeud 0 de l'arbre de branchement, nous avons pensé procéder à l'élimination de variables avant le noeud 0. Tel que décrit à la sous-section 1.2.3, toutes les variables ayant un coût réduit supérieur ou égal à la différence entre la valeur de la solution entière trouvée dans la partie heuristique et la valeur de la solution duale de la relaxation linéaire obtenue au noeud 0 (i.e. $\bar{c}_{ij}^k \geq Z_{IP} - Z_{LP}$) peuvent être éliminées du problème puisqu'elles ne peuvent faire partie de la solution optimale entière. En se reportant à ce critère, il est possible de constater que trois éléments sont nécessaires pour procéder à l'élimination de variables : une borne inférieure, une solution duale réalisable pour la relaxation linéaire, et une borne supérieure. Il importe maintenant de s'attarder un peu plus à ces trois éléments pour spécifier leur rôle et définir un parallèle entre les données utilisées par HADJAR ET AL. (2003) et celles utilisées ici.

(a) Premièrement, une borne inférieure est nécessaire. Dans la méthode de HADJAR ET AL. (2003), la procédure d'élimination des variables est faite après la résolution de la relaxation linéaire au noeud 0 de l'arbre de branchement. La borne inférieure utilisée est donc la valeur de la solution duale de la relaxation linéaire obtenue au noeud 0 de l'arbre de branchement (Z_{LP}). Dans notre cas, la borne inférieure consiste plutôt en la valeur optimale du PTVDU ; cette valeur est trouvée par l'optimiseur commercial à l'étape (ii) de la méthode heuristique.

(b) La solution duale réalisable pour la relaxation linéaire, notée $\bar{\pi}$, est nécessaire pour calculer les coûts réduits des variables. Pour HADJAR ET AL. (2003), la solution duale est la solution obtenue par la résolution de la relaxation linéaire au noeud 0 de l'arbre de branchement. Dans notre cas, il s'agit en fait de la solution duale obtenue comme suit.

En associant respectivement les variables duales β_i, ω et γ_i aux contraintes (3.2), (3.3) et (3.4), le problème dual associé au PTVDU (3.1)-(3.5) s'écrit :

$$\text{Maximiser} \quad \omega \sum_{k \in K} r_k + \sum_{i \in N} \beta_i \tag{3.6}$$

sujet à

$$\omega + \gamma_j \leq d_{Sj} \quad \forall (S, j) \in E \tag{3.7}$$

$$\beta_i + \gamma_j - \gamma_i \leq d_{ij} \quad \forall (i, j) \in E \; i \neq S, j \neq S \tag{3.8}$$

$$\beta_i - \gamma_i \leq d_{iS} \quad \forall (i, S) \in E \tag{3.9}$$

$$\omega \leq 0. \tag{3.10}$$

De la même façon, en associant les variables duales λ_i, μ_k et π_i^k aux contraintes (1.2), (1.3) et (1.4) du PTVDM, il est possible de formuler le problème dual associé

au PTVDM (1.1)-(1.5) :

$$\text{Maximiser} \sum_{k \in K} \mu_k r_k + \sum_{i \in N} \lambda_i \tag{3.11}$$

sujet à

$$\mu_k + \pi_j^k \leq c_{n+k,j} \quad \forall (n+k, j, k) \in A \tag{3.12}$$

$$\lambda_i + \pi_j^k - \pi_i^k \leq c_{ij} \quad \forall (i, j, k) \in A \quad i \neq n+k, j \neq n+k \tag{3.13}$$

$$\lambda_i - \pi_i^k \leq c_{i,n+k} \quad \forall (i, n+k, k) \in A \tag{3.14}$$

$$\mu_k \leq 0 \quad \forall k \in K. \tag{3.15}$$

Soit $(\bar{\beta}, \bar{\omega}, \bar{\gamma})$, une solution réalisable pour le problème dual du PTVDU. Soit $(\bar{\lambda}, \bar{\mu}, \bar{\pi})$, une solution obtenue comme suit :

$$\bar{\lambda}_i = \bar{\beta}_i \quad \forall i \in N \tag{3.16}$$

$$\bar{\mu}_k = \bar{\omega} \quad \forall k \in K \tag{3.17}$$

$$\bar{\pi}_i^k = \bar{\gamma}_i \quad \forall k \in K, \ \forall i \in N. \tag{3.18}$$

À ce point, il s'agit de vérifier que la solution $(\bar{\lambda}, \bar{\mu}, \bar{\pi})$ est réalisable pour le problème dual du PTVDM. En fait, pour un arc intertâche $(i, j, k) \in A$, on a :

$$\bar{\lambda}_i + \bar{\pi}_j^k - \bar{\pi}_i^k = \bar{\beta}_i + \bar{\gamma}_j - \bar{\gamma}_i \quad \leq d_{ij} = c_{ij}. \tag{3.19}$$

Pour un arc $(n+k, j, k) \in A$, on a :

$$\bar{\mu}_k + \bar{\pi}_j^k = \bar{\omega} + \bar{\gamma}_j \quad \leq d_{Sj} \leq c_{n+k,j}. \tag{3.20}$$

Pour un arc $(i, n+k, k) \in A$, on a :

$$\bar{\lambda}_i - \bar{\pi}_i^k = \bar{\beta}_i - \bar{\gamma}_i \quad \leq d_{iS} \leq c_{i,n+k}. \tag{3.21}$$

Enfin, l'inégalité $\bar{\mu}_k \leq 0$ est satisfaite.

Cette démonstration montre qu'une solution duale du PTVDM obtenue à partir d'une solution duale réalisable du PTVDU est nécessairement réalisable pour le problème dual du PTVDM. Cette dernière est donc utilisée pour calculer les coûts réduits des variables.

(c) Le troisième élément utile est une borne supérieure. Dans le cas de HADJAR ET AL., la valeur de la solution réalisable entière trouvée dans la partie heuristique (Z_{IP}) agit à titre de borne supérieure. Dans le cas présent, la borne supérieure est plutôt égale à la valeur de la solution obtenue pour le PTVDM à la troisième étape de la méthode heuristique.

Après avoir calculé la différence entre la borne supérieure et la borne inférieure, il a été possible de constater que cette différence est toujours beaucoup plus élevée que le plus élevé des coûts réduits. Dans ce cas, le critère proposé par HADJAR ET AL. (2003) ne peut permettre l'élimination de variables. Il faut donc conclure que, malgré la pertinence de cette idée, il a été impossible de la mettre en application étant donné que les résultats produits par la méthode heuristique manquaient de précision pour l'utilisation que nous voulions en faire.

La solution réalisable obtenue grâce à la méthode heuristique décrite précédemment permettra donc seulement d'établir une partition initiale pour débuter l'agrégation dynamique de contraintes. Dans la partition, un élément consiste en une tournée, i.e. un ensemble de tâches. Il faut ici noter que seules les tâches formant les tournées sont conservées pour former la partition initiale pour l'agrégation dynamique de contraintes, les dépôts auxquels sont affectées les tournées n'étant pas pris en compte dans la partition. Les résultats présentés à la prochaine section montrent que la partition initiale obtenue à partir de la solution heuristique n'est pas de très bonne qualité puisque peu de combinaisons de tâches de la partition initiale ont été conservées dans la solution de la relaxation linéaire.

Bien que l'agrégation dynamique de contraintes soit décrite à la section 1.3.2, il im-

porte d'en spécifier ici deux idées directement liées : le concept d'*incompatibilité* et le processus de résolution *multi-phases*. Le vocable incompatibilité est utilisé pour désigner une suite de tâches différant de la partition courante. Par exemple, si un élément de la partition courante correspond à la suite de tâches T_1, T_8, T_{14}, T_{32} et que, lors de la résolution des sous-problèmes, une des suites de tâches générées soit T_1, T_8, T_{14}, T_{35}, alors il est possible de dire qu'il y a une incompatibilité entre la partition courante (T_{14}, T_{32}) et la suite de tâches générée (T_{14}, T_{35}). Les lignes qui suivent expliquent comment le concept d'incompatibilité est géré au cours de la résolution.

Dans le présent cas, chaque tâche à accomplir est représentée par un noeud $i = 1, 2, ..., n$ dans le graphe $G = (V, E)$. À chacun des noeuds représentant une tâche est associé son prédécesseur et son successeur dans la partition courante. Les premières (resp. dernières) tâches des tournées n'ont pas de prédécesseurs (resp. successeurs) parce que, comme il a été mentionné précédemment, seules les tâches sont considérées dans la formation de la partition initiale, et non les dépôts. Une *ressource* est utilisée pour calculer le nombre d'incompatibilité lors de la résolution des sous-problèmes. Pour chaque tâche, une borne inférieure, égale à 0, et une borne supérieure déterminent l'intervalle à l'intérieur duquel le nombre d'incompatibilités comprises dans une tournée doit se situer pour que la dite tournée soit valide et puisse être générée. Si le nombre d'incompatibilités cumulées dans une tournée est supérieur à la borne supérieure de l'intervalle, alors aucune colonne ne peut être générée à partir de cette tournée. Une incompatibilité supplémentaire est comptabilisée lors de la construction d'une tournée dès que le prédécesseur ou le successeur d'une tâche comprise dans la tournée est non-cohérente avec la partition courante.

Le principe de résolution multi-phases permet, quant à lui, d'augmenter la borne supérieure de l'intervalle utilisé pour gérer la ressource d'incompatibilité, et ce au fur et à mesure des itérations de génération de colonnes. Le changement de phase s'effectue lorsqu'il n'est plus possible de générer des colonnes de coût réduit négatif

à une itération donnée. Pour permettre de retrouver rapidement la partition initiale, la borne supérieure est fixée à 0 pour débuter ; cela signifie qu'aucun écart n'est permis par rapport à la partition courante, qui est en l'occurrence la partition initiale. Dans le cas présent, les phases permises sont 0, 1, 2 et ∞. La borne supérieure doit augmenter jusqu'à une limite ∞ d'incompatibilités permises pour assurer l'exactitude de la méthode.

3.2 Les attentes et les résultats

Deux impacts sont attendus de l'agrégation dynamique de contraintes en regard des résultats présentés dans l'article de ELHALLAOUI ET AL. (2003). Premièrement, une réduction du temps de résolution des relaxations linéaires rencontrées dans tous les noeuds de l'arbre de branchement. Également, une diminution du nombre de variables fractionnaires contenues dans les solutions de ces relaxations linéaires. Cette étude débute par l'exploration des gains potentiels au noeud 0 de l'arbre de branchement.

Après avoir observé le comportement de quelques instances de différentes tailles, il a été possible de conclure que, à la fin de la relaxation linéaire, la plupart des instances sont presque complètement désagrégées, c'est-à-dire que le nombre d'ensembles de tâches est pratiquement égal au nombre de tâches comprises dans l'instance à solutionner. À la lumière de cette observation, les paramètres ont été ajustés pour effectuer rapidement la désagrégation, sans pour autant que celle-ci soit trop abrupte. En effet, une désagrégation trop abrupte aurait pour conséquence néfaste d'augmenter trop rapidement la taille du problème maître, et d'en ralentir la résolution. Cela irait donc à l'encontre d'un des objectifs de l'agrégation dynamique de contraintes, c'est-à-dire diminuer le temps de résolution du problème maître. L'agrégation n'est donc permise qu'une seule fois dans tout le processus, i.e. au commencement lors de la constitution de la partition initiale.

Tableau 3.1 – Tableau comparatif entre la méthode de HADJAR ET AL. (2003) sans agrégation dynamique de contraintes et la méthode de HADJAR ET AL. (2003) avec agrégation dynamique de contraintes

	Hadjar et al.(2003)				Hadjar et al.(2003) - avec Agreg.Dyn.				
taille	**tRL(s)**	**tSP(s)**	**tPM(s)**	**NbIt**	**tRL(s)**	**tSP(s)**	**tPM(s)**	**Am.tRL(%)**	**NbIt**
300t-4d	50	20	30	85	35	24	10	30,9	66
400t-4d	145	53	93	112	89	58	32	38,5	80
500t-4d	321	109	211	146	152	109	43	52,5	93
800t-4d	2055	571	1484	286	666	440	225	67,6	133
300t-6d	48	22	26	62	46	37	9	5,4	67
400t-6d	130	56	74	78	108	85	23	16,7	80
500t-6d	253	101	152	90	198	159	40	21,6	94
800t-6d	1604	465	1139	155	858	698	160	46,5	143

Le tableau 3.1 présente des résultats comparatifs pour la méthode de HADJAR ET AL. (2003) lorsque celle-ci est utilisée sans et avec agrégation dynamique de contraintes. Dans les deux cas, seule la relaxation linéaire au noeud 0 a été résolue. Le temps de la relaxation linéaire au noeud 0 (tRL) est donc donné, ainsi que le détail du temps consacré à la résolution des sous-problèmes (tSP) et du problème maître (tPM). Le nombre d'itérations nécessaires pour résoudre le noeud 0 (NbIt) est également fourni. Pour la méthode de HADJAR ET AL. (2003) avec agrégation dynamique de contraintes, un pourcentage d'amélioration du temps de résolution de la relaxation linéaire au noeud 0 (Am.tRL) est calculé par rapport à la méthode sans agrégation dynamique de contraintes. Les résultats présentés sont des moyennes calculées sur cinq instances du même type de problème (même nombre de tâches et de dépôts). Tous les temps de calcul sont exprimés en secondes.

Les résultats du tableau 3.1 montrent que les améliorations les plus importantes du temps de calcul de la relaxation linéaire au noeud 0 sont observées pour les instances ayant quatre dépôts. En fait, pour les instances comptant quatre dépôts et 300, 400, 500 et 800 tâches, les pourcentages moyens d'amélioration du temps

de calcul de la relaxation linéaire au noeud 0 sont respectivement de 30,9%, 38,5%, 52,5% et 67,6%. Pour les instances comptant le même nombre de dépôts, plus la taille du problème augmente en termes de nombre de tâches, plus les pourcentages d'amélioration ont tendance à augmenter. Les améliorations sont meilleures pour les instances comportant seulement quatre dépôts parce que la solution heuristique générée selon la méthode énoncée à la section précédente est plus précise lorsque le nombre de dépôts est petit, notamment à cause de la façon dont les tournées sont affectées aux dépôts lors de l'étape (iii) du processus heuristique. Aussi, il est possible de constater que, dans la méthode avec agrégation dynamique de contraintes, le temps passé à résoudre le problème maître est beaucoup plus court que dans la méthode sans agrégation dynamique de contraintes. Par contre, le temps passé à résoudre les sous-problèmes tend à augmenter considérablement entre la méthode sans agrégation dynamique de contraintes et la méthode avec agrégation dynamique de contraintes, et ce pour des problèmes de même taille. Cela est occasionné par l'ajout d'une ressource permettant de gérer le nombre d'incompatibilités comprises dans une tournée dans la version avec agrégation dynamique de contraintes ; ce qui a pour impact d'augmenter de façon non-négligeable le temps de résolution des sous-problèmes. Pour un même type de problèmes, le nombre d'itérations est soit plus petit, soit légèrement plus élevé dans la méthode avec agrégation dynamique de contraintes. Également, pour les instances ayant le même nombre de tâches, celles avec six dépôts prennent toujours moins d'itérations pour résoudre le noeud 0 que celles avec quatre dépôts lorsque la méthode de résolution traditionnelle est utilisée. Cela est dû au fait que le nombre de sous-problèmes est plus élevé dans un problème comptant six dépôts et n tâches que dans un problème comptant quatre dépôts et n tâches. Le nombre de colonnes pouvant être générées à chaque itération est donc plus grand dans le premier cas. Cependant, cette tendance n'est pas aussi marquée dans le cas avec agrégation dynamique de contraintes, et cela est probablement attribuable à l'ajout d'une ressource dont il faut tenir compte dans les sous-problèmes de cette

version. En fait, comme la majeure partie du temps est passé à résoudre les sous-problèmes dans la version avec agrégation dynamique de contraintes, les instances comptant six dépôts sont nécessairement plus longues à résoudre que celles avec quatre dépôts puisqu'elles comptent deux sous-problèmes supplémentaires. Pour un même type de problème, le nombre de variables fractionnaires contenues dans la solution de la relaxation linéaire est sensiblement le même peu importe la version utilisée (avec ou sans agrégation dynamique de contraintes).

Aucun test n'a été effectué pour tenter de résoudre les problèmes à l'optimalité en utilisant cette deuxième méthode de résolution étant donné que la solution de la relaxation linéaire est généralement complètement désagrégée. Cela ne laisse donc pas présager de gains potentiels en termes de temps de calcul pour les problèmes en nombres entiers.

Après avoir effectué plusieurs tests avec des configurations de paramètres différentes, il a été possible de constater que les instances générées selon le modèle de CARPANETO ET AL. (1989) ne sont pas idéales pour expérimenter l'agrégation dynamique de contraintes. En fait, ces instances se caractérisent par des tournées courtes, c'est-à-dire qui contiennent peu de tâches (en moyenne trois ou quatre tâches par tournée). Ces problèmes sont donc très peu dégénérés. Enfin, les tailles des problèmes testés sont trop petites pour constater l'efficacité de la résolution avec agrégation dynamique de contraintes.

Pour terminer cette deuxième et dernière partie sur les stratégies de résolution, il est possible d'avancer les conclusions suivantes :

(a) Le temps de résolution de la relaxation linéaire au noeud 0 de l'arbre de branchement a été réduit dans certains cas en intégrant la technique d'agrégation dynamique de contraintes à la méthode de HADJAR ET AL. (2003) pour la résolution des instances de CARPANETO ET AL. (1989).

(b) Pour des problèmes comportant le même nombre de dépôts, plus le nombre de tâches augmente, plus les améliorations en termes de temps de calcul de la relaxation linéaire sont grandes. Les améliorations les plus importantes ont été notées pour les problèmes comportant quatre dépôts, probablement à cause de la qualité de la solution heuristique permettant de générer la partition initiale.

(c) Puisque la solution de la relaxation linéaire obtenue au noeud 0 de l'arbre de branchement est presque totalement désagrégée, cela ne laisse entrevoir qu'un faible potentiel d'accélération pour la résolution complète de l'arbre de branchement. Cela justifie qu'aucun test n'a été fait pour résoudre les problèmes à l'optimalité en nombres entiers.

CONCLUSION

Le problème de tournées de véhicules avec dépôts multiples est un problème *NP-difficile.* La seule méthode exacte connue pouvant résoudre ce problème à l'optimalité demeure l'exploration d'un arbre de branchement en appliquant le principe de séparation et d'évaluation progressive. Dans un tel contexte, le développement de stratégies de résolution pour résoudre le problème de tournées de véhicules à l'optimalité est donc pertinent.

Deux stratégies ont été présentées dans ce mémoire. La première utilise une méthode heuristique pour trouver une solution réalisable au problème de tournées de véhicules avec dépôts multiples. En utilisant un critère d'élimination des variables basé sur le coût réduit, il est possible, si la solution réalisable obtenue est de bonne qualité, d'éliminer un pourcentage élevé de variables avant de débuter l'exploration de la partie exacte de l'arbre de branchement. Cela permet, entre autres, de réduire la taille du graphe sur lequel il faut résoudre les sous-problèmes lors de la recherche de la solution optimale. Cette méthode a révélé que, au-delà d'un certain pourcentage de variables éliminées, généralement 90%, le fait d'éliminer plus de variables n'entraîne pas nécessairement une diminution du temps de calcul pour la recherche de la solution optimale dans la partie exacte, et inversement.

La seconde stratégie proposée utilise l'agrégation dynamique de contraintes pour résoudre le problème de tournées de véhicules avec dépôts multiples. Cette stratégie concourt à l'atteinte de deux objectifs. Le premier objectif vise à réduire les temps de calcul des relaxations linéaires rencontrées dans l'arbre de branchement. Le second objectif a pour but de diminuer le nombre de variables fractionnaires dans les solutions obtenues de ces relaxations linéaires. Après maintes expérimentations sur les

instances de CARPANETO ET AL. (1989), il a été possible de conclure que le temps de calcul de la relaxation linéaire peut être diminué en intégrant la méthode d'agrégation dynamique de contraintes à la métode de HAJDAR ET AL. (2003). Les instances composées d'un nombre élevé de tâches sont celles qui présentent les meilleures améliorations en termes de temps de résolution de la relaxation linéaire au noeud 0 de l'arbre de branchement. Par contre, étant donné que la solution de la relaxation linéaire obtenue au noeud 0 de l'arbre de branchement est presque totalement désagrégée, cela ne laisse entrevoir qu'un faible potentiel d'accélération pour la résolution en nombres entiers.

Ce mémoire lance des pistes à d'autres chercheurs qui pourraient vouloir se pencher sur le problème de tournées de véhicules avec dépôts multiples pour y développer des stratégies de résolution et d'accélération. L'agrégation dynamique de contraintes est en fait une méthode qui laisse entrevoir un grand potentiel d'accélération si elle débute avec une bonne partition initiale. D'autres chercheurs pourraient donc s'intéresser à développer des méthodes heuristiques qui fourniraient des solutions initiales de meilleure qualité que celle proposée dans ce mémoire. Également, comme l'agrégation dynamique de contraintes est particulièrement efficace pour résoudre les problèmes qui présentent beaucoup de dégénérescence, alors il peut devenir pertinent d'effectuer les tests sur des instances présentant cette caractéristique. Conjointement à l'agrégation dynamique de contraintes, l'exploration d'un arbre de branchement heuristique pourrait être favorisée au détriment d'un arbre de branchement exhaustif, la taille de ce dernier pouvant être démesurée pour les instances comportant un nombre élevé de tâches et de dépôts, instances pourtant intéressantes pour tester l'agrégation dynamique de contraintes.

Les modifications récemment apportées à la version de la méthode d'agrégation dynamique de contraintes utilisée dans le cadre de ce mémoire nous permettent de croire que nos résultats auraient pu être différents, et peut-être même meilleurs, si la nouvelle version avait été disponible au moment de notre recherche.

BIBLIOGRAPHIE

[1] BERTOSSI, A. A., CARRARESI, P. et GALLO, G. (1987). On Some Matching Problems Arising in Vehicle Scheduling Models. *Networks*, **17**, 271-281.

[2] BIANCO, L., MINGOZZI, A. et RICCIARDELLI, S. (1994). A Set Partitioning Approach to the Multiple Depot Vehicle Scheduling Problem. *Optimization Methods and Software*, **3**, 163-194.

[3] CARPANETO, G., DELL'AMICO, M., FISCHETTI, M. et TOTH, P. (1989). A Branch and Bound Algorithm for the Multiple Depot Vehicle Scheduling Problem. *Networks*, **19**, 531-548.

[4] DADUNA, J. R. et MOJSILOVIC, M. (1988). Computer-Aided Vehicle and Duty Scheduling Using the HOT Programme System. In : J. R. Daduna, A. Wren (Eds.), Computer-Aided Transit Scheduling. *Lecture Notes in Economics and Mathematical Systems*, **308**, Hamburg, Germany, 133-146.

[5] DANTZIG, G.B. et WOLFE, P. (1960). Decomposition Principle for Linear Programs. *Operations Research*, **8**, 101-111.

[6] DELL'AMICO, M., FISCHETTI, M. et TOTH, P. (1993). Heuristic Algorithms for the Multiple Depot Vehicle Scheduling Problem. *Management Science*, **39**, 115-125.

[7] DESAULNIERS G., LAVIGNE, J. et SOUMIS, F. (1998). Multi-Depot Vehicle Scheduling Problems with Time Windows and Waiting Costs. *European Journal of Operational Research*, **111**, 479-494.

[8] ELHALLAOUI, I., VILLENEUVE, D., SOUMIS, F. et DESAULNIERS, G. (2003). Dynamic Aggregation of Set Partitioning Constraints in Column Generation.

Les Cahiers du GERAD, **G-2003-45**, HEC Montréal, Canada. À paraître dans *Operations Research.*

[9] FISCHETTI, M., LODI, A., MARTELLO, S. et TOTH, P. (2001). A Polyhedral Approach to Simplified Crew Scheduling and Vehicle Scheduling Problems. *Management Science*, **47**, 833-850.

[10] FORBES, M. A., HOLT, J. N. et WATTS, A. M. (1994). An Exact Algorithm for Multiple Depot Bus Scheduling. *European Journal of Operational Research*, **72**, 115-124.

[11] HADJAR, A., MARCOTTE, O. et SOUMIS, F. (2003). A Branch-and-Cut Algorithm for the Multiple Depot Vehicle Scheduling Problem. *Les Cahiers du GERAD*, **G-2001-25**, HEC Montréal, Canada. À paraître dans *Operations Research.*

[12] LAVIGNE, J. (1996). Le Problème de Tournées de Véhicules avec Fenêtres de Temps et Dépôts Multiples. *Mémoire de Maîtrise*, École Polytechnique de Montréal, Canada.

[13] LÖBEL, A. (1998). Vehicle Scheduling in Public Transit and Lagrangean Pricing. *Management Science*, **44**, 1637-1649.

[14] MESQUITA, M. et PAIXÃO, J. (1992). Multiple Depot Vehicle Scheduling Problem : A New Heuristic Based on Quasi-Assignment Algorithms. In : M. Desrochers, J.-M. Rousseau (Eds.), Computer-Aided Transit Scheduling. *Lecture Notes in Economics and Mathematical Systems*, **386**, Springer, Berlin, 167-180.

[15] MESQUITA, M. et PAIXÃO, J. (1994). A Multiplier Adjustment Method for the Multi-Depot Vehicle Scheduling Problem. *Triennal Symposium on Transportation Analysis*, Capri, Italy, Juin 1994, 749-762.

[16] MINGOZZI, A., BIANCO, L. et RICCIARDELLI, S. (1995). An Exact Algorithm for Combining Vehicle Trips. In : J. R. Daduna, I. Branco, J. Paixão

(Eds.), Computer-Aided Transit Scheduling. *Lecture Notes in Economics and Mathematical Systems*, **430**, Springer, Berlin, 145-172.

[17] OUKIL, A. (2004). Problème de Tournées de Véhicules à Horizon Long : étude numérique d'une approche de stabilisation proximale. *Mémoire de Maîtrise*, École Polytechnique de Montréal, Canada.

[18] RIBEIRO, C. C. et SOUMIS, F. (1994). A Column Generation Approach to the Multiple Depot Vehicle Scheduling Problem. *Operations Research*, **42**, 41-52.